UNIVERZA V LJUBLJANI

FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Robert Lampreht

Analiza algoritmov linearnega programiranja

DIPLOMSKO DELO

UNIVERZITETNI ŠTUDIJSKI PROGRAM
PRVE STOPNJE RAČUNALNIŠTVO IN INFORMATIKA

Ljubljana, 2014

UNIVERZA V LJUBLJANI

FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Robert Lampreht

Analiza algoritmov linearnega programiranja

DIPLOMSKO DELO

UNIVERZITETNI ŠTUDIJSKI PROGRAM
PRVE STOPNJE RAČUNALNIŠTVO IN INFORMATIKA

MENTOR: doc. dr. Tomaž Dobravec

Ljubljana, 2014

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

Linearno programiranje je optimizacijski problem iskanja optimuma linearne kriterijske funkcije, pri čemer mora rešitev zadoščati linearnim omejitvam. Problem linearnega programiranja je zelo pomemben, saj se nanj lahko prevedejo številni drugi optimizacijski problemi.
V diplomskem delu preglejte in predstavite področje linearnega programiranja. Opišite nekatere znane pristope k reševanju tega problema (metoda simpleksov ter metode notranjih točk) in predstavite nekaj konkretnih algoritmov za reševanje.
V praktičnem delu naloge sprogramirajte predstavljene algoritme v programskem jeziku C#. Algoritme izvedite na standardnih testnih podatkih in predstavite rezultate testiranja.

IZJAVA O AVTORSTVU DIPLOMSKEGA DELA

Spodaj podpisani Robert Lampreht, z vpisno številko **63010082**, sem avtor diplomskega dela z naslovom:

Analiza algoritmov linearnega programiranja

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom doc. dr. Tomaža Dobravca,
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela,
- soglašam z javno objavo elektronske oblike diplomskega dela na svetovnem spletu preko univerzitetnega spletnega arhiva.

V Ljubljani, dne 25. septembra 2014 Podpis avtorja:

Zahvalil bi se svoji ženi Ivani, ker je moje sonce kadar je oblačno ter svojim staršem in družini za podporo. Še posebej pa se zahvaljujem mojemu mentorju doc. dr. Tomažu dobravcu za potrpežljivost in ves dodaten čas, ki mi ga je namenil.

Kazalo

Povzetek

Uporaba linearnega programiranja je danes zelo razširjena saj se lahko nanj prevedejo številni problemi. Nekatere od pomembnejših industrijskih panog, ki uporabljajo optimizacijske motode so: logistika, telekomunikacija in predelovalne dejavnosti.
Cilj linearnega programiranja je optimizacija kompleksnih problemov z namenskimi algoritmi. V diplomski nalogi želimo predstaviti različne tipe algoritmov za optimizacijo ter njihovo delovanje. Osredotočili se bomo predvsem na dva različni metodi: metodo simpleksov in metodo notranje točke. Za testiranje bomo standardne optimizacijeske testne množice s katerimi bomo prišli do rezultatov. Dobljene rezultate bomo analizirali in poskušali utemeljiti kateri način optimizacijskega postopka je učinkovitejši.

Ključne besede: linearno programiranje, optimizacijske metode in algoritmi, metoda simpleksov, metoda notranjih točk

Abstract

The use of linearprogramming today is very wide spread because of useful optimizations problems that can solve. Some of the more important industries that are using optimization methods are: logistics, telecommunications and manufacturing.
The objective of linear programming is optimization of complex problems with algorithms. In out thesis we want to introduce different types of optimization algorithms and how they work. Our main focus will be on simplex method and interior-point method. For testing purposes we will use standardized test set for optimization problems. With given results we will try to decide which optimization method is better suited for the job.

Keywords: linear programming, optimization methods and algorithms, simplex method, interior-point method

Poglavje 1 Uvod

Linearno programiranje je področje matematike, ki se ukvarja s problemom optimizacije z omejitvami. Predstavlja specifičen razred optimizacijskih problemov, kjer maksimiziramo (minimiziramo) linearno funkcijo, pri čemer upoštevamo linearne omejitve. Pobudnik razvoja linearnega programiranja (1930) je Leonid Kantorovic, z metodo reševanja problema planiranja proizvodnje. Za začetnika linearnega programiranja velja George B. Dantzig, ki je leta 1947 razvil metodo simpleksov, pomembno vlogo pa ima tudi John von Neumann, ki je istega leta postavil temelje teorije dualnosti [1, 2].

V Združenih državah Amerike se je linearno programiranje razvilo med drugo svetovno vojno z namenom, da reši zapletene probleme načrtovanja logistike vojaških operacijah. Doprinos k razvoju linearnega programiranja prištevamo ekonomistu Tjalling Koopmansu (rojen na Nizozemskem leta 1940, pozneje preseljen v Združene države Amerike). Matematik Kantorovic in ekonomist Koopmans sta leta 1975 dobila Nobelovo nagrado za ekonomijo - za prispevke k teoriji optimalne izrabe sredstev, kjer je linearno programiranje igralo glavno vlogo. Veliko industrijskih podjetij uporablja linearno programiranje kot standardno orodje (npr. za optimalno raporejanje končnih sredstev).

Gre torej za zelo pogosto uporabljeno metodo pri reševanju optimizacijskih problemov z omejitvami. V samem postopku ločimo tri pomembne korake, in sicer [1, 3]:

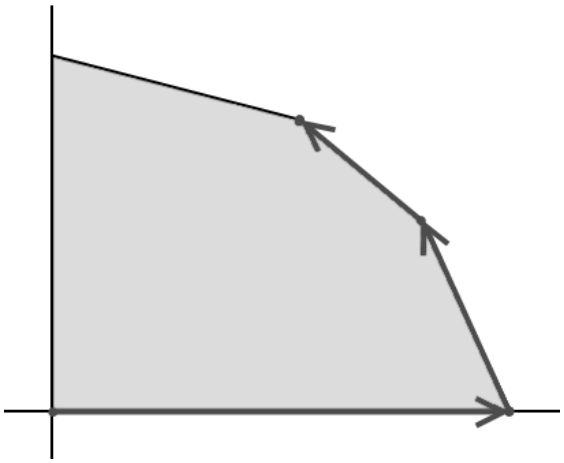
- formulacija problema (postavitev problema v pravilni obliki);
- rešitev (izračun optimalne možnosti);
- analiza občutljivosti (kaj bi se zgodilo, če bi se pogoji našega zastavljenega problema malo spremenili).

Linearno programiranje je metoda za iskanje optimalne kriterijske funkcije, ko so omejitve (in kriterijska funkcija) dane v obliki sistema linearnih neenačb. Rešitev takšnega sistema je mogoče dobiti v grafični obliki, vendar le v primeru, ko imamo zgolj dve ali tri odločitvene spremenljivke, sicer pa je potreben računski postopek. Splošna računska metoda za reševanje linearnega programiranja je metoda simpleksov, ki jo je leta 1984 razvil metematik George B. Dantzig [1].

Poglavje 2 Metoda simpleksov

Kljub temu, da je bilo že pred tem veliko napisanega o linearnem programiranju, štejemo šele leto 1947 za pravi začetek linearnega programiranja. Tega leta je namreč George B. Dantzig odkril metodo simpleksov in od takrat naprej je linearno programiranje doživelo velik vzpon ter se močno razširilo na najrazličnejša področja znanosti in gospodarstva [4].

Metoda simpleksov je standardna tehnika pri reševanju linearnega programiranja, kjer nastopajo tri ali celo več odločitvenih spremenljivk. Da bi lahko rešili takšne probleme potrebujemo algebraično interpretacijo postopka iskanja optimalne rešitve. Vse v praksi uporabljane metode so iterativne: začnemo z neko možno bazno rešitvijo, ki jo postopoma izboljšujemo. Teoretično možno, a neracionalno bi bilo enostavno poiskati vse možne bazne rešitve ter med njimi izbrati robno vrednost, kar bi pomenilo izračun sistema m enačb z n neznankami. Simpleks metoda gre s sistematskimi koraki od ene do druge bazne rešitve tako, da je vrednost kriterijske funkcije Z v vsakem koraku bližja optimalni vrednosti. Gibanje od začetne do končne rešitve poteka vzdolž enega roba konveksnega poliedra množice možnih rešitev, v vsaki robni točki ugotavljamo, ali je to optimalna rešitev in če ni, kam moramo naprej. Če pridemo v robno točko, iz katere gre rob v neskončnost in če Z ni omejena, potem je tudi rešitev neomejena [5].



Slika 1: Primer poteka iskanja optimalne rešitve s simpleksnim algoritmom.

Velja pravilo, da je robna točka optimalna, kadar nobena sosednja robna točka ne da boljše rešitve. Kljub temu, da je robnih točk končno mnogo, obstajajo primeri, kjer je lahko število robnih točk eksponentno (npr. n-dimenzionalna hiperkocka)

2.1 Grafična (geometrijska) predstavitev

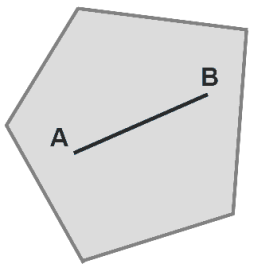
Z grafično metodo ugotovimo lastnosti problemov linearnega programiranja, ki so osnova za vse analitične metode [5]:

- množica možnih rešitev M je konveksna;
- optimalna rešitev je v eni ali več robnih točkah množice M;
- konveksni polieder M ima končno mnogo robnih točk;
- imamo končno množico v kateri iščemo optimalno rešitev;
- ni potrebno pregledati vseh robnih točk, da bi prišli do optimalne rešitve.

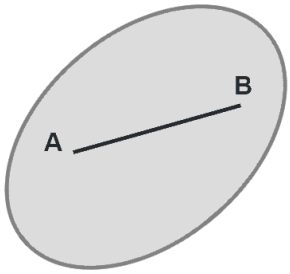
Linearni program s spremenljivkama x in y je predstavljen s funkcijo $Z = c_1x + c_2y$, ki jo imenujemo kriterijska funkcija in mora biti optimizirana znotraj sistema podanih neenačb. Spremenljivki x in y se imenujeta strukturni spremenljivki, rešitev sistema neenačb pa množica možnih rešitev.

Pri vsakem linearnem programu z dvema spremenljivkama, mora biti množica možnih rešitev konveksna množica točk (Slika 2-3), katera mora imeti naslednje lastnosti [9]:

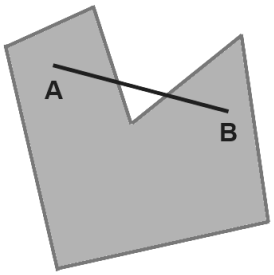
- mejo množice sestavlja končno število premic ali njihovih segmentov
- za vsak par točk A in B v množici, se mora v množici v celoti nahajati tudi daljica, ki jih povezuje



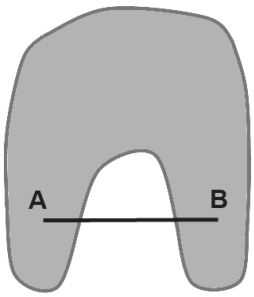
Slika 2: Konveksna množica.



Slika 3: Konveksna množica.



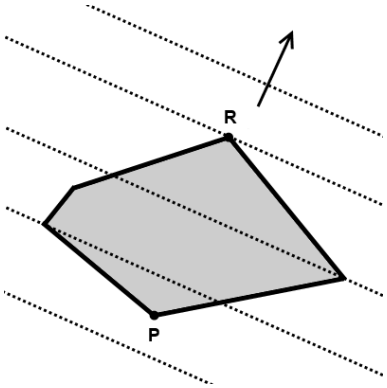
Slika 4: Nekonveksna množica.



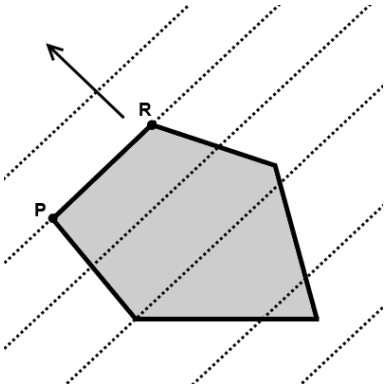
Slika 5: Nekonveksna množica.

Pri reševanju linearnega programa, se iz celotnega sklopa možnih rešitev izbere tista, ki da optimalno rešitev (maksimum ali minimum) kriterijske funkcije. Pri programih, ki vključujejo le dve spremenljivki, je možna grafična predstavitev množice možnih rešitev. Običajno ta množica vsebuje neskončno število točk, razen v primerih, kadar so za rešitev zahtevane celoštevilske strukturne spremenljivke

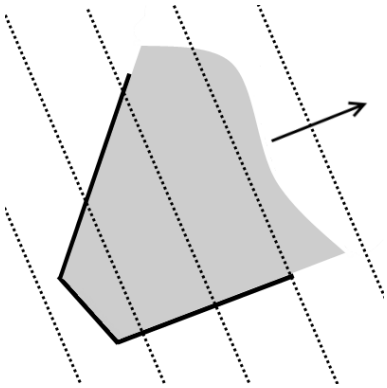
Za vsako konstanto C v kriterijski funkciji Z, kjer velja Z = C, dobimo izolinijo. Izolinije so premice, ki sestavljajo ravnino, katero določa kriterijska funkcija Z. Vse izolinije imajo enak nagib. Z večanjem konstante C, dobimo smer naraščanja izolinije. Naša optimalna rešitev leži na zadnji največji možni izoliniji, ki še vsebuje rešitev iz množice možnih rešitev



Slika 6: Maksimum v R, minimum v P.



Slika 7: Maksimum vzdolž premice PR.



Slika 8: Maksimum ne obstaja.

2.2 Problem pivovarja

Za boljšo predstavo reševanja linearnih problemov, si bomo ogledali problem pivovarja, pri katerem bomo uporabili grafično predstavitev simpleksne metode.

Imamo majhno pivovarno, katera prideluje svetlo in temno pivo. Proizvodnja je omejena z naslednjimi surovinami: koruzo, hmeljem in ječmenovim sladom. Recept za svetlo in temno pivo se razlikuje v količini potrebnih surovin za pripravo. Surovine in količina, ki jih imamo na voljo, so podane v tabeli (Tabela 1). V tej tabeli imamo tudi podatke, koliko surovin potrebujemo za en sod temnega in en sod svetlega piva ter kakšen je profit od prodaje [8].

	koruza (kg)	**hmelj (dkg)**	**slad (kg)**	**profit (€)**
na razpolago	480	160	1190	
temno (1 sod)	5	4	35	13
svetlo (1 sod)	15	4	20	23

Tabela 1

Izbrati želimo kombinacijo, s katero bi porabili čim več surovin in naredili čim več profita. V naslednji tabeli (Tabela 2) imamo nekaj veljavnih kombinacij, pri katerih ne presegamo razpoložljivih surovin. Kombinacije števila sodov s temnim in svetlim pivom so izbrane po občutku, mi pa želimo optimalno rešitev. Zanima nas, ali obstaja rešitev, pri kateri je dobiček večji kot pri tej, ki smo jo uganili.

	koruza (kg)	**hmelj (dgk)**	**slad (kg)**	**profit (€)**
samo temno (34 sodov)	179	136	**1190**	442
samo svetlo (32 sodov)	**480**	128	640	736
20 sodov temno, 20 sodov svetlo	400	**160**	1100	720
bolj profitne kombinacije?	?	?	?	>736 ?

Tabela 2

1. **kombinacija:** odločimo se za izdelavo samo temnega piva. Pridelamo lahko maksimalno 34 sodov, saj pri tem porabimo celotno razpoložljivo zalogo slada. Ostane nam 310 kg koruze in 24 dkg hmelja. Naredimo 442 € profita.
2. **kombinacija:** odločimo se za izdelavo izključno svetlega piva. V tem primeru pridelamo samo 32 sodov. Ostane nam 32 dkg hmelja in 550 kg slada. Porabimo pa celotno zalogo koruze. Kljub manjšemu številu sodov vseeno naredimo večji dobiček, in sicer 736 €.
3. **kombinacija:** tokrat poskusimo z izdelavo obeh vrst piva. Pridelamo 20 sodov obojega. Ostanek pri tem je 80 kg koruze in 90 kg sladu, hmelja ne ostane nič. Dobiček znaša 720 €. Torej manj kot v primeru pridelave samo svetlega piva.

Matematična formulacija problema

A – število sodov temnega piva

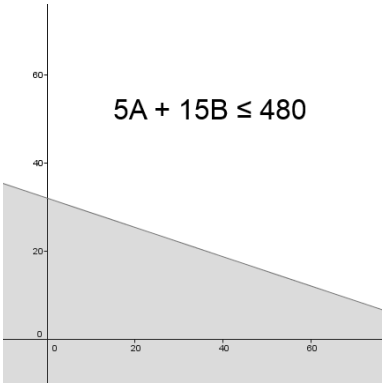
B – število sodov svetlega piva

	temno		**svetlo**			
maximum	**13A**	**+**	**23B**	**=**	**Z**	**profit**
omejitev 1	5A	+	15B	≤	480	koruza
omejitev 2	4A	+	4B	≤	160	hmelj
omejitev 3	35A	+	20B	≤	1190	slad
omejitev 4			A	≥	0	
			B	≥	0	

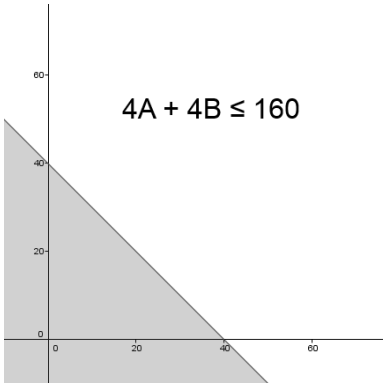
Tabela 2

Število sodov ne more biti negativno, zato smo dodali še dva dodatna pogoja, ki omejita število sodov na pozitivna števila.

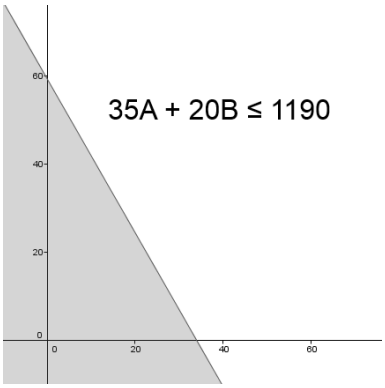
Vse neenakosti je potrebno prikazati v koordinatnem sistemu (slika 9–12):



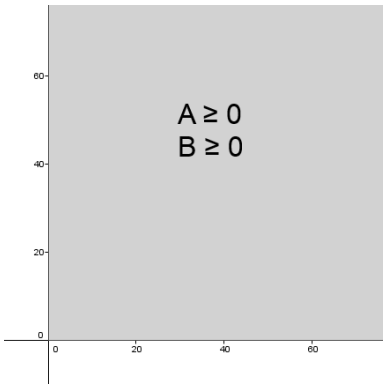
Slika 9: Omejitev 1.



Slika 10: Omejitev 2.

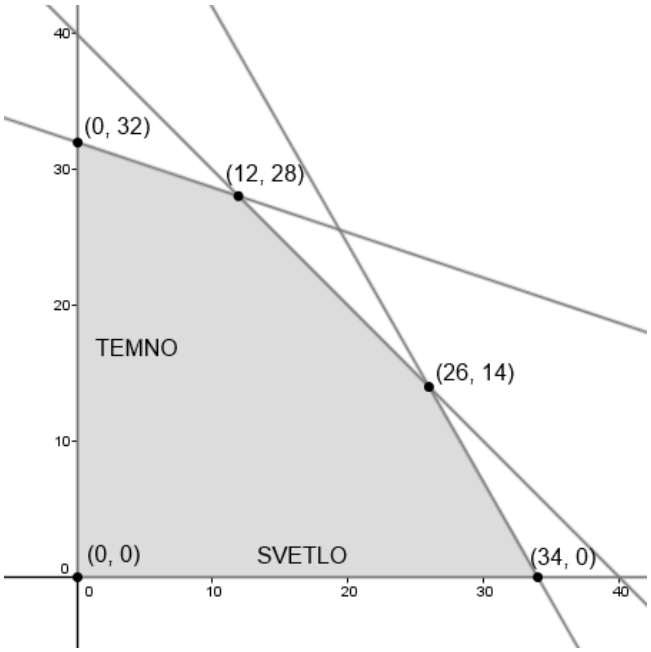


Slika 11: Omejitev 3.



Slika 12: Omejitev 4.

Ko združimo vse neenakosti dobimo množico možnih rešitev (slika 13):



Slika 13: Množica možnih rešitev.

Simpleksna metoda narekuje, da je rešitev problema vedno v eni od robnih točk. V našem primeru dobimo pet takšnih točk: (0,0), (34,0), (26,14), (12,28) in (0,32). Točko (0,0) lahko izločimo, ker je očitna rešitev 0. Točki (34,0) in (0,32) smo že izračunali (1. in 2. kombinacija). Ostaneta nam še dve točki:

- **(26,14):** 26 sodov temnega in 14 sodov svetlega piva => 660 € profita
- **(12,28):** 12 sodov temnega in 28 sodov svetlega piva => 800 € profita

Kombinacija s katero naredimo največ profita je 12 sodov temnega in 28 sodov svetlega piva.

Poglavje 3 Metode notranjih točk

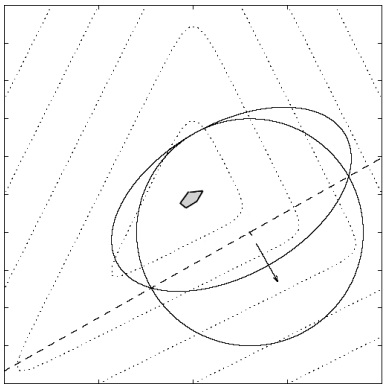
3.1 Elipsoidna metoda

Leta 1978 je ruski matematik L.G. Khachiyan s pomočjo elipsoidne metode dokazal, da sodi linearno programiranje v razred problemov s polinomsko časovno zahtevnostjo. Elipsoidno metodo so sicer leta 1972 iznašli Naum Z. Shor, Arkady Nemirovsky in David B. Yudin ter jo uporabljali za reševanje konveksnih problemov. Elipsoidna metoda je za linearno programiranje v teoriji zelo pomembna. Do takrat je bila to edina metoda, katera je imela dokazano polinomsko časovno zahtevnost. Čeprav je v teoriji hitra pa se v praksi izkaže za prepočasno, da bi bila uporabna.

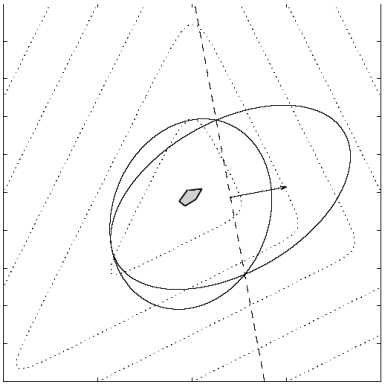
Delovanje algoritma [7]:

1. Najdi elipsoid, ki sigurno vsebuje vse rešitve sistema.
2. Če je center ellipsoida veljavna rešitev: vrni zadetek!
3. Če je volumen elipsoida premajhen: vrni nerešljivo!
4. Če je vrednost izven veljavnega območja, razpolovi elipsoid tako, da so vse možne rešitve v eni od polovic
5. Ustvari nov elipsoid, ki pokrije polovico prejšnjega elipsoida z rešitvami in pojdi na 2. korak.

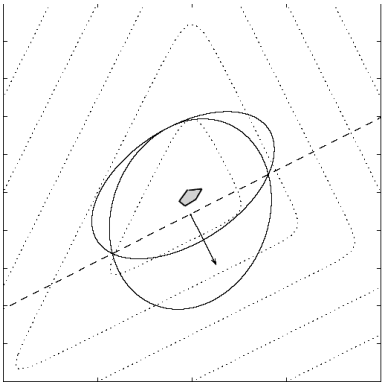
Na slikah 14-19 je grafično prikazan potek postopka elipsoidne metode po korakih.



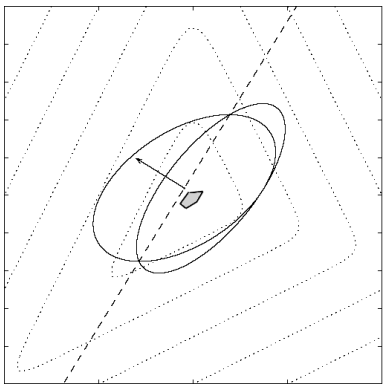
Slika 14: Prvi korak.



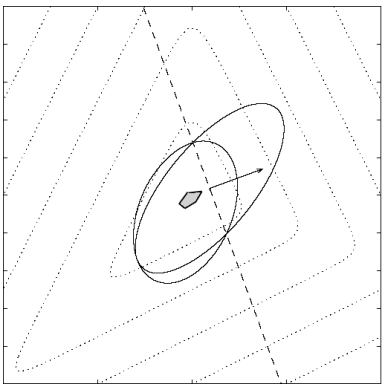
Slika 15: Drugi korak.



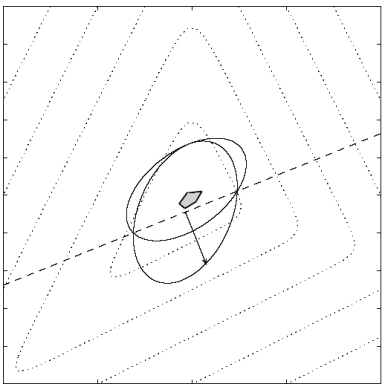
Slika 16: Tretji korak.



Slika 17: Četrti korak.



Slika 18: Peti korak.



Slika 19: Šesti korak.

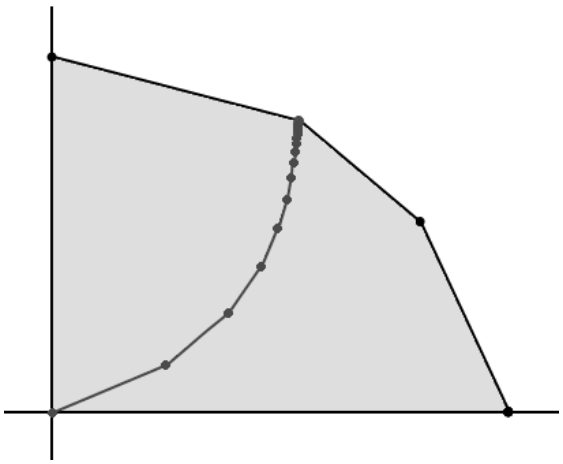
Elipsoidna metoda sicer ne spada pod kategorijo metod notranjih točk, vendar jo je vseeno vredno omeniti saj je bila osnova za enega izmed prvih popularnejših algoritmov, ki je uporabljal metodo notranjih točk: Karmarkar-jev algoritem.

3.2 Iskanje poti z metodo notranjih točk

Metodo notranjih točk je prvič omenil Von Neumann, ko je predlagal novo metodo za linearno programiranje, katera je uporabljala Gordanov homogeni linearni sistem. Leta 1984 je bila ta metoda popularizirana s Karmarkarjevim algoritmom, ki ga je razvil indijski matematik Narendra Karmarkar, za linearno programiranje. Z njim je uspel dokazati polinomsko časovno zahtevnost reševanja linearnih problemov [13, 14].

Ta metoda ne skače po robu dopustnega območja kakor metoda simpleksov, ampak izbira dopustne rešitve v njegovi notranjosti. Za razliko od elipsoidne metode je Karmarkarjev algoritem precej uporaben tudi v praktičnih linearnih problemih. Primer uporabe je rešitev kompleksnega problema v mreži komunikacij, kjer so čas reševanja z nekaj tednov zmanjšali na nekaj dni.

Leta 1985 v IBM-u in AT&T-ju »izumijo« afino skaliranje (A. S.), različica Karmarkarjevega algoritma z intuitivnim iskanjem. V začetnih računskih testih je A.S. prekosila simpleksni in Karmarkarjev algoritem. Leta 1989 so ugotovili, da je A.S. algoritem že izumil Dikin leta 1967 [20].



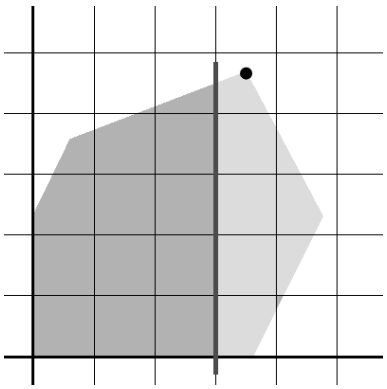
Slika 20: Prikaz iskanja rešitve z metodo notranjih točk.

3.3 Metoda rezanja ravnine

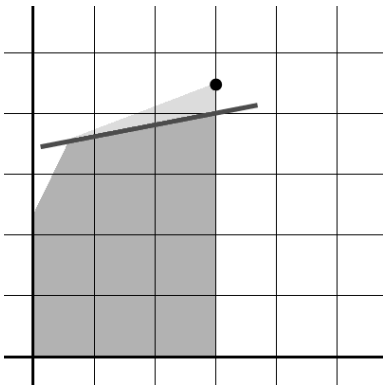
V matematični optimizaciji pojem metode rezanja ravnin pokriva več različnih metod, ki delujejo po principu iterativnega določanja natančnejše množice rešitev z uporabo neenakosti (množice manj optimalnih rešitev reže stran). Takšne procedure se uporabljajo za iskanje celoštevilskih rešitev, kot tudi za rešitve mešanih števil. Metodo rezanja ravnin je v petdesetih letih predstavil Ralph Gomory kot metodo za reševanje mešanih in celoštevilskih programskih problemov. Večina strokovnjakov, vključno z Gomory-jem samim, so menili, da je metoda nepraktična zaradi numerične nestabilnosti ter tudi neučinkovita zaradi velikega števila rezov in zaokroževanj, ki so bila potrebna, da je bil narejen napredek proti rešitvi. Stvari so se obrnile, ko je v sredini devetdesetih let Cornuejols s sodelavci dokazal, da je metoda v kombinaciji z metodo »razveji in omeji« zelo učinkovita in lahko premaga numerično nestabilnost. Nova metoda je poimenovana »razveji in odreži«.
Danes se v vseh komercialnih aplikacijah za reševanje problemov z mešanimi števili uporablja takšna ali drugačna različica Gomory-jeve metode. S pomočjo generiranja iz simpleksnih tabel je ta metoda zelo učinkovita [15].

Metoda rezanja ravnine v primeru problema z mešanimi števili deluje tako, da najprej reši necelоštevilski linearni program. Po teoriji, ki jo narekuje linearno programiranje, predvideva (če so zadoščeni vsi potrebni pogoji), da lahko vedno najde optimum v robni točki. Ta optimum se preveri, če je celoštevilski. Če ni, potem zagotovo obstaja linearna neenakost, ki loči ta optimum od konveksne množice možnih rešitev. Iskanje takšne neenakosti se imenuje ločitveni problem, neenakost pa rez. Ta rez na določenem delu konveksne množice omeji del rešitve na celo število. Ta postopek se nadaljuje, dokler ni najdena optimalna celoštevilska rešitev.

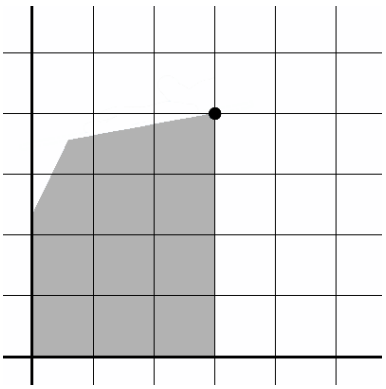
Primer rezanja ravnine: iskanje maksimuma s celoštevilskimi rešitvami, kot je prikazano na slikah 21, 22 in 23.



Slika 21: Prvi korak.



Slika 22: Drugi korak.



Slika 23: Tretji korak.

Poglavje 4 Algoritmi

V tem poglavju si bomo podrobneje pogledali naslednje algoritme s praktičnimi računskimi primeri:

- simpleksni algoritem
- Gomory-jev algoritem rezanja ravnine
- Karmarkar-jev algoritem

4.1 Simpleksni algoritem

Simpleksna tabela

Simpleksna metoda izvaja elementarne transformacije nad vrsticami matrike, ki jo imenujemo simpleksna tabela. Ta tabela je sestavljena iz kriterijske funkcije, omejitvenih enačb in njihovih vrednosti.

A – matrika koeficientov podanih linearnih omejitev
b – matrika vrednosti, ki se nahajajo na desni strani enačb linearnih omejitev
c – matrika koeficientov kriterijske funkcije
Z – trenutna vrednost kriterijske funkcije (maksimum ali minimum, ki ga iščemo)

$A_{(m\times n)}$	$b_{(m\times 1)}$
$c_{(1\times n)}$	Z

Tabela 3: Struktura simpleksne tabele.

Pivotni postopek

Za izboljšanje trenutne rešitve moramo vpeljati novo osnovno spremenljivko, imenujemo jo vstopajoča spremenljivka. Kar pomeni, da moramo eno od osnovnih spremenjivk odstraniti – odhajajoča spremenljivka. Vstopajočo in odhajajočo spremenljivko izberemo na naslednji način:

1. Vstopajoča spremenljivka ustreza najmanjši (najbolj negativni) vrednosti v zadnji vrstici tabele.
2. Odhajajoča spremenljivka ustreza najmanjšemu nenegativnemu razmerju b_i/a_{ij} v stolpcu, kjer se nahaja vstopajoča spremenljivka
3. Vrednost v tabeli, ki se nahaja v stolpcu vstopajoče in v vrstici odhajajoče spremenljivke, se imenuje pivot.

Za tvorjenje izboljšane rešitve uporabimo Gauss-Jordanovo eliminacijsko metodo nad stolpcem, ki vsebuje pivot.

Primer:

Podano imamo kriterijsko funkcjo: $Z = 6x + 5y + 4z$

in omejitve: $2x + y + z \leq 180$

$x + 3y + 2z \leq 300$

$2x + y + 2z \leq 240$

$x, y, z \geq 0$

Izračunati želimo maksimum kriterijske funkcije (max Z).

1. neenakosti pretvorimo v enakosti tako, da vpeljemo dopolnilne spremenljivke S_1, S_2 in S_3.

$$2x + y + z + S_1 = 180$$

$$x + 3y + 2z + S_2 = 300$$

$$2x + y + 2z + S_3 = 240$$

2. preoblikujemo kriterijsko funkcijo

   -6x - 5y -4z + Z = 0

3. simpleksne tabele

Koeficiente linearnih enačb in kriterijske funckcije vpišemo v simpleksno tabelo

x	**y**	**z**	**S_1**	**S_2**	**S_3**	**Z**	**C**
2	1	1	1	0	0	0	180
1	3	2	0	1	0	0	300
2	1	2	0	0	1	0	240
–6	–5	–4	0	0	0	1	0

Določiti moramo pivot:

- pivot stolpec določa najnižja vrednost v zadnji vrstici;
- pivot vrstico določa najnižja vrednost delitelja desne strani enačbe in vrednosti v pivot stolpcu (180/2 = 90, 300/1 = 300, 240/2 = 120);
- pivot = 2.

x	**y**	**z**	**S_1**	**S_2**	**S_3**	**Z**	**C**
2	1	1	1	0	0	0	180
1	3	2	0	1	0	0	300
2	1	2	0	0	1	0	240
–6	–5	–4	0	0	0	1	0

Pivot moramo spraviti na 1:

- prvo vrstico pomnožimo z $^1/_2$, da dobimo pivot enak 1;
- ostale vrstice prepišemo.

x	y	z	S_1	S_2	S_3	Z	C
1	$^1/_2$	$^1/_2$	$^1/_2$	0	0	0	90
1	3	2	0	1	0	0	300
2	1	2	0	0	1	0	240
–6	–5	–4	0	0	0	1	0

Ostale vrednosti v pivot stolpcu moramo spraviti na 0:

- prvo vrstico prepišemo;
- prvo vrstico odštejemo od druge vrstice, da dobimo prvo vrednost v drugi vrstici enako 0;
- prvo vrstico pomnožimo z 2 in jo odštejemo od tretje vrstice;
- sedaj imamo v prvem stolpcu samo eno vrednost 1 in vse ostale 0;
- poiščemo pivot po enakem postopku kot v 1. koraku;
- pivot = $^5/_2$.

x	y	z	S_1	S_2	S_3	Z	C
1	$^1/_2$	$^1/_2$	$^1/_2$	0	0	0	90
0	$\mathbf{^5/_2}$	$^3/_2$	$-^1/_2$	1	0	0	210
0	0	1	-1	0	1	0	60
0	–2	–1	3	0	0	1	540

Enako kot v 2. koraku moramo pivot spraviti na 1:

- drugo vrstica pomnožimo z $^2/_5$;
- ostale vrstice prepišemo.

x	y	z	s1	s2	s3	Z	C

1	$^1/_2$	$^1/_2$	$^1/_2$	0	0	0	90
0	**1**	$^3/_5$	$-^1/_5$	$^2/_5$	0	0	84
0	0	1	–1	0	1	0	60
0	–2	–1	3	0	0	1	540

Kakor v 3. koraku, moramo ostale vrednosti v pivot stolpcu spraviti na 0:

- drugo vrstico pomnožimo z $^1/_2$ in jo odštejemo od prve;
- tretja vrstica že ima vrednost 0 v pivot stolpcu, zato ni potrebno narediti nič;
- zadnja vrstica ne vsebuje nobene negativne vrednosti, kar pomeni, da je postopek končan.

x	**y**	**z**	**S_1**	**S_2**	**S_3**	**Z**	**C**
1	0	$^1/_5$	$^3/_5$	$-^1/_5$	0	0	48
0	1	$^3/_5$	$-^1/_5$	$^2/_5$	0	0	84
0	0	1	–1	0	1	0	60
0	0	$^1/_5$	$^{13}/_5$	$^4/_5$	0	1	**708**

MAXIMUM

Rešitev

V poštev pridejo samo stolpci, ki vsebujejo eno vrednost 1, ostale pa 0. Druge stolpce zanemarimo. Rešitve preberemo tako, da za vsaki spremenljivki (na vrhu tabele) v veljavnem stolpcu dodelimo vrednost C iz tiste vrstice, kjer je vrednost 1. Ostale spremenljivke so 0. Vrednosti, ki jih dobimo, so naslednje:

x = 48, y = 84, z = 0, s1 = 0, s2 = 0, s3 = 60 in Z = 708
Maksimum kriterijske funkcije Z = 6x + 5y + 4z je 708.

Če poskusimo vstaviti vrednosti, da se prepričamo:

- kriterijska funkcija: 6*48 + 5*84 = 708 ✓
- 1. enačba: 2*48 + 84 = 180 ✓
- 2. enačba: 48 + 3*84 = 300 ✓
- 3. enačba: 2*48 + 84 + 60 = 240 ✓

4.2 Gomory-jev algoritem rezanja ravnine

Gomory-jev algoritem rezanja ravnine se uporablja takrat, kadar želimo, da je naša rešitev celo število. V realnem svetu so takšne zahteve dokaj pogoste. Dostikrat je rešitev uporabna samo v primeru, če je celo število. Praktičen primer takšne zahteve je kadrovanje. Kadar razporejamo ljudi, bo vedno govora o celih številih.

Podano imamo kriterijsko funkcjo [17]: $Z = 2x + 15y + 18z$

in omejitve: $-x + 2y - 6z \leq -10$
$y + 2z \leq 6$
$2x + 10z \leq 19$
$-x + y \leq -2$
$x, y, z \geq 0$

Izračunati želimo minimum kriterijske funkcije (min Z). Ta problem lahko rešimo z dualnim simpleksnim algoritmom. Dobimo naslednjo končno tabelo:

	x	**y**	**z**	$\mathbf{S_1}$	$\mathbf{S_2}$	$\mathbf{S_3}$	$\mathbf{S_4}$	**C**
x	1	10	0	5	0	3	0	7
y	0	5	0	2	1	1	0	5
z	0	–2	1	–1	0	$-1/2$	0	$1/2$
$\mathbf{S_4}$	0	11	0	5	0	3	1	5
	0	31	0	8	0	3	0	–23

Katera ustreza rešitvam: $x = 7$, $y = 0$, $z = 1/2$ } $Z = 23$

Vendar to še ni končna rešitev, saj morajo x, y in z biti cela števila. Spremeniti moramo tretjo vrstico:

$$-2y + z - S_1 - {}^1\!/_2 S_3 = {}^1\!/_2$$

$$-2y + z - S_1 - \lfloor {}^1\!/_2 \rfloor S_3 \leq {}^1\!/_2$$

$$-2y + z - S_1 - S_3 \leq {}^1\!/_2$$

Če so vse spremenljivke cela števila, potem velja tudi:

$$-2y + z - S_1 - S_3 \leq \lfloor {}^1\!/_2 \rfloor$$

$$-2y + z - S_1 - S_3 \leq 0$$

To novo neenakost lahko dodamo obstoječi tabeli v obliki:

$$-2y + z - S_1 - S_3 - S_5 = 0$$

Z vnosom nove neenakosti dobimo naslednjo tabelo:

	x	**y**	**z**	$\mathbf{S_1}$	$\mathbf{S_2}$	$\mathbf{S_3}$	$\mathbf{S_4}$	$\mathbf{S_5}$	**C**
x	1	10	0	5	0	3	0	0	7
y	0	5	0	2	1	1	0	0	5
z	0	–2	1	–1	0	$-{}^1\!/_2$	0	0	${}^1\!/_2$
$\mathbf{S_4}$	0	11	0	5	0	3	1	0	5
$\mathbf{S_5}$	0	–2	1	–1	0	–1	0	1	0
	0	31	0	8	0	3	0	0	–23

Matriko koeficientov v tabeli označujemo z $A_B^{-1}A$. Torej tisti stolpci, ki ustrezajo osnovnim spremenljivkam morajo dati rešitev $A_B^{-1}A_B = I$. Če pogledamo označene stolpce, vidimo, da temu ni tako v stolpcu z. To popravimo tako, da zadnji vrstici (S_5) odštejemo vrednost tretje vrstice:

	x	**y**	**z**	**S_1**	**S_2**	**S_3**	**S_4**	**S_5**	**C**
x	1	10	0	5	0	3	0	0	7
y	0	5	0	2	1	1	0	0	5
z	0	–2	1	–1	0	$-\frac{1}{2}$	0	0	$\frac{1}{2}$
S_4	0	11	0	5	0	3	1	0	5
S_5	0	0	0	0	0	$-\frac{1}{2}$	0	1	$-\frac{1}{2}$
	0	31	0	8	0	3	0	0	–23

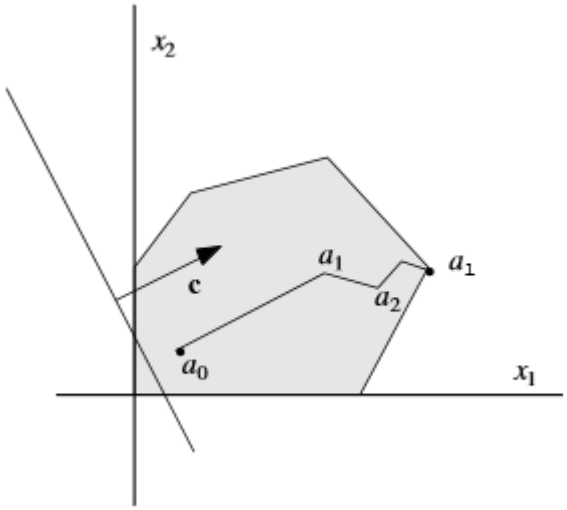
Izberemo pivot vrednost po simpleksnem pravilu. Ta se nahaja v stolpcu S_3 in vrstici S_5. Po pivotiranju dobimo:

	x	**y**	**z**	**S_1**	**S_2**	**S_3**	**S_4**	**S_5**	**C**
x	1	10	0	5	0	0	0	6	4
y	0	5	0	2	1	0	0	2	4
z	0	–2	1	–1	0	0	0	–1	1
S_4	0	11	0	5	0	0	1	6	2
S_5	0	0	0	0	0	1	0	–2	1
	0	31	0	8	0	0	0	6	–26

Ta rešitev da optimalno celoštevilčno rešitev: $x = 4$, $y = 0$, $z = 1$ $\Rightarrow Z = 26$

4.3 Karmarkar-jev algoritem

Za razliko od simpleksne metode si zaporedja iteracij Karmarkar-jevega algoritma ni tako lahko vizualno predstavljati. Namesto premikanja iz ene robne točke na sosednjo, kjer se z vsakim premikom približamo optimalni rešitvi, nas Karmarkar vodi z zavezanimi očmi. Uporablja mnogo transformacij, s katerimi nam zamegli jasno predstavo o tem, kaj se dogaja s prvotnim problemom.
Osnova za delovanje tega algoritma je naslednja: če izhajamo nekje iz sredine poligonske množice možnih rešitev, moramo iti v »smeri najstrmejšega spusta« proti meji in bomo prišli dokaj hitro do rešitve [18].



Slika 24: Primer iskalne poti Karmarkar-jevega algoritma.

Linearni problem mora biti podan v naslednji obliki [19]:

$$\text{Min } Z = C^T X$$

kjer velja:
$$AX = 0$$
$$1X = 1$$
$$X \geq 0$$

$$X = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \qquad C = \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{bmatrix} \qquad 1 = [1\ 1\ \dots\ 1]_{(1\times n)} \qquad A = \begin{bmatrix} c_{11} & c_{12} & \dots & c_{1n} \\ c_{21} & c_{22} & \dots & c_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ c_{m1} & c_{m2} & \dots & c_{mn} \end{bmatrix} \qquad n \geq 2$$

Predvidevamo tudi, da je $X_0 = \left[1/n \quad 1/n \quad \dots \quad 1/n\right]^T$ veljavna rešitev in $Z_{min} = 0$.

Postopek izvajanja Karmarkar-jevega algoritma

1. korak:

- Izračunamo $r = \frac{1}{\sqrt{n\,(n-1)}}$ in $\alpha = \frac{n-1}{3n}$
- Nastavimo k = 0 in $X_0 = \left[1/n \quad 1/n \quad \dots \quad 1/n\right]^T$

2. korak:

- $D_0 = \mathrm{diag}(X_0)$
- $\overline{C} = C^T D_0$
- $P = \begin{bmatrix} AD_0 \\ 1 \end{bmatrix}$
- Izračunamo $C_P = \left[I - P^T(PP^T)^{-1}P\right] \overline{C}^T$

Če je $C_P = 0$, smo dobili optimalno rešitev, algoritem se konča. V nasprotnem primeru pa se algoritem nadaljuje:

- $Y_{nov} = X_0 - \alpha r \frac{C_P}{\|C_P\|}$
- $X_{k+1} = \frac{D_k Y_{nov}}{1 D_k Y_{nov}}$
- Dokažemo lahko, da za k = 0 velja $\frac{D_k Y_{nov}}{1 D_k Y_{nov}} = Y_{nov}$ torej $X_1 = Y_{nov}$
- $Z = C^T X_{k+1}$

Povečamo k za ena (k = k +1) in ponovimo 2. korak. To ponavljamo dokler vrednost kriterijske funkcije ni enaka 0 ali manj.

Primer:

Podano imamo kriterijsko funkcjo: Z = 2y – z

in omejitve: x – 2y + z = 0
x + y + z = 1
x, y, z ≥ 0

Izračunati želimo minimum kriterijske funkcije (min Z).

1. iteracija (k = 0)

$$D_0 = \operatorname{diag}(X_0) = \begin{bmatrix} 1/3 & 0 & 0 \\ 0 & 1/3 & 0 \\ 0 & 0 & 1/3 \end{bmatrix}$$

$$\overline{C} = C^T D_0 = [0 \quad 2 \quad -1] \times \begin{bmatrix} 1/3 & 0 & 0 \\ 0 & 1/3 & 0 \\ 0 & 0 & 1/3 \end{bmatrix} = [0 \quad 2/3 \quad -1/3]$$

$$AD_0 = [1 \quad -2 \quad 1] \times \begin{bmatrix} 1/3 & 0 & 0 \\ 0 & 1/3 & 0 \\ 0 & 0 & 1/3 \end{bmatrix} = [1/3 \quad -2/3 \quad 1/3]$$

$$P = \begin{bmatrix} AD_0 \\ 1 \end{bmatrix} = \begin{bmatrix} 1/3 & -2/3 & 1/3 \\ 1 & 1 & 1 \end{bmatrix}$$

$$PP^T = \begin{bmatrix} 1/3 & -2/3 & 1/3 \\ 1 & 1 & 1 \end{bmatrix} \times \begin{bmatrix} 1/3 & 1 \\ -2/3 & 1 \\ 1/3 & 1 \end{bmatrix} = \begin{bmatrix} 2/3 & 0 \\ 0 & 3 \end{bmatrix}$$

$$(PP^T)^{-1} = \begin{bmatrix} 3/2 & 0 \\ 0 & 1/3 \end{bmatrix}$$

$$P^T(PP^T)^{-1}P = \begin{bmatrix} 1/3 & 1 \\ -2/3 & 1 \\ 1/3 & 1 \end{bmatrix} \times \begin{bmatrix} 3/2 & 0 \\ 0 & 1/3 \end{bmatrix} \times \begin{bmatrix} 1/3 & -2/3 & 1/3 \\ 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 1/2 & 0 & 1/2 \\ 0 & 1 & 0 \\ 1/2 & 0 & 1/2 \end{bmatrix}$$

$$C_P = [I - P^T(PP^T)^{-1}P]\,\overline{C}^T = \left(\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} - \begin{bmatrix} 1/2 & 0 & 1/2 \\ 0 & 1 & 0 \\ 1/2 & 0 & 1/2 \end{bmatrix} \right) \times \begin{bmatrix} 0 \\ 2/3 \\ -1/3 \end{bmatrix} = \begin{bmatrix} 1/6 \\ 0 \\ -1/6 \end{bmatrix}$$

$$\|C_P\| = \sqrt{(1/6)^2 + 0 + (-1/6)^2} = \sqrt{\frac{2}{36}} = \frac{\sqrt{2}}{6}$$

$$Y_{nov} = X_0 - \alpha r \frac{C_P}{\|C_P\|} = \begin{bmatrix} 1/3 \\ 1/3 \\ 1/3 \end{bmatrix} - \left(\frac{\frac{2}{9} \times \frac{1}{\sqrt{6}}}{\frac{\sqrt{2}}{6}} \right) \times \begin{bmatrix} 1/6 \\ 0 \\ -1/6 \end{bmatrix} = \begin{bmatrix} 0{,}2692 \\ 0{,}3333 \\ 0{,}3974 \end{bmatrix}$$

$$X_1 = Y_{nov} = \begin{bmatrix} 0{,}2692 \\ 0{,}3333 \\ 0{,}3974 \end{bmatrix}$$

$$Z = C^T X_1 = [0 \quad 2 \quad -1] \times \begin{bmatrix} 0{,}2692 \\ 0{,}3333 \\ 0{,}3974 \end{bmatrix} = \mathbf{0{,}2692}$$

Ker je $Z > 0$ nadaljujemo.

2. iteracija (k = 1)

$$D_1 = \mathrm{diag}(X_1) = \begin{bmatrix} 0{,}2692 & 0 & 0 \\ 0 & 0{,}3333 & 0 \\ 0 & 0 & 0{,}3974 \end{bmatrix}$$

$$\overline{C} = C^T D_1 = [0 \quad 2 \quad -1] \times \begin{bmatrix} 0{,}2692 & 0 & 0 \\ 0 & 0{,}3333 & 0 \\ 0 & 0 & 0{,}3974 \end{bmatrix} = [0 \quad 0{,}6667 \quad -0{,}3974]$$

$$AD_1 = [1 \quad -2 \quad 1] \times \begin{bmatrix} 0{,}2692 & 0 & 0 \\ 0 & 0{,}3333 & 0 \\ 0 & 0 & 0{,}3974 \end{bmatrix} = [0{,}2692 \quad -0{,}6667 \quad 0{,}3974]$$

$$P = \begin{bmatrix} AD_1 \\ 1 \end{bmatrix} = \begin{bmatrix} 0{,}2692 & -0{,}6667 & 0{,}3974 \\ 1 & 1 & 1 \end{bmatrix}$$

$$PP^T = \begin{bmatrix} 0{,}2692 & -0{,}6667 & 0{,}3974 \\ 1 & 1 & 1 \end{bmatrix} \times \begin{bmatrix} 0{,}2692 & 1 \\ -0{,}6667 & 1 \\ 0{,}3974 & 1 \end{bmatrix} = \begin{bmatrix} 0{,}675 & 0 \\ 0 & 3 \end{bmatrix}$$

$$(PP^T)^{-1} = \begin{bmatrix} 1{,}482 & 0 \\ 0 & 0{,}333 \end{bmatrix}$$

$$P^T(PP^T)^{-1}P = \begin{bmatrix} 0{,}2092 & 1 \\ -0{,}6668 & 1 \\ 0{,}4574 & 1 \end{bmatrix} \times \begin{bmatrix} 1{,}4335 & 0 \\ 0 & 0{,}3333 \end{bmatrix} \times \begin{bmatrix} 0{,}2092 & -0{,}6668 & 0{,}4574 \\ 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 0{,}441 & 0{,}067 & 0{,}492 \\ 0{,}067 & 0{,}992 & -0{,}059 \\ 0{,}492 & -0{,}059 & 0{,}567 \end{bmatrix}$$

$$C_P = [I - P^T(PP^T)^{-1}P]\,\overline{C}^T = \left(\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} - \begin{bmatrix} 0{,}441 & 0{,}067 & 0{,}492 \\ 0{,}067 & 0{,}992 & -0{,}059 \\ 0{,}492 & -0{,}059 & 0{,}567 \end{bmatrix} \right) \times \begin{bmatrix} 0 \\ 0{,}6667 \\ -0{,}3974 \end{bmatrix} = \begin{bmatrix} 0{,}151 \\ -0{,}018 \\ -0{,}132 \end{bmatrix}$$

$$\|C_P\| = \sqrt{(0{,}151)^2 + (-0{,}018)^2 + (-0{,}132)^2} = 0{,}2014$$

$$Y_{nov} = X_0 - \alpha r \frac{C_P}{\|C_P\|} = \begin{bmatrix} 1/3 \\ 1/3 \\ 1/3 \end{bmatrix} - \left(\frac{2}{9} \times \frac{1}{\sqrt{6}} \over 0{,}2014 \right) \times \begin{bmatrix} 0{,}151 \\ -0{,}018 \\ -0{,}132 \end{bmatrix} = \begin{bmatrix} 0{,}2653 \\ 0{,}3414 \\ 0{,}3928 \end{bmatrix}$$

$$D_1 Y_{nov} = \begin{bmatrix} 0{,}2692 & 0 & 0 \\ 0 & 0{,}3333 & 0 \\ 0 & 0 & 0{,}3974 \end{bmatrix} \times \begin{bmatrix} 0{,}2653 \\ 0{,}3414 \\ 0{,}3928 \end{bmatrix} = \begin{bmatrix} 0{,}0714 \\ 0{,}1138 \\ 0{,}1561 \end{bmatrix}$$

$$1D_1 Y_{nov} = [1 \quad 1 \quad 1] \times \begin{bmatrix} 0{,}0714 \\ 0{,}1138 \\ 0{,}1561 \end{bmatrix} = 0{,}3413$$

$$X_2 = \frac{D_1 Y_{nov}}{1 D_1 Y_{nov}} = \frac{1}{0{,}3413} \times \begin{bmatrix} 0{,}0714 \\ 0{,}1138 \\ 0{,}1561 \end{bmatrix} = \begin{bmatrix} 0{,}2092 \\ 0{,}3334 \\ 0{,}4574 \end{bmatrix}$$

$$Z = C^T X_2 = [0 \quad 2 \quad -1] \times \begin{bmatrix} 0{,}2092 \\ 0{,}3334 \\ 0{,}4574 \end{bmatrix} = \mathbf{0{,}2094}$$

Ker je Z > 0 nadaljujemo.

3. iteracija (k = 2)

$$D_2 = \mathrm{diag}(X_2) = \begin{bmatrix} 0{,}2092 & 0 & 0 \\ 0 & 0{,}3334 & 0 \\ 0 & 0 & 0{,}4574 \end{bmatrix}$$

$$\overline{C} = C^T D_2 = [0 \quad 2 \quad -1] \times \begin{bmatrix} 0{,}2092 & 0 & 0 \\ 0 & 0{,}3334 & 0 \\ 0 & 0 & 0{,}4574 \end{bmatrix} = [0 \quad 0{,}6668 \quad -0{,}4574]$$

$$AD_2 = [1 \quad -2 \quad 1] \times \begin{bmatrix} 0{,}2092 & 0 & 0 \\ 0 & 0{,}3334 & 0 \\ 0 & 0 & 0{,}4574 \end{bmatrix} = [0{,}2092 \quad -0{,}6668 \quad 0{,}4574]$$

$$P = \begin{bmatrix} AD_2 \\ 1 \end{bmatrix} = \begin{bmatrix} 0{,}2092 & -0{,}6668 & 0{,}4574 \\ 1 & 1 & 1 \end{bmatrix}$$

$$PP^T = \begin{bmatrix} 0{,}2092 & -0{,}6668 & 0{,}4574 \\ 1 & 1 & 1 \end{bmatrix} \times \begin{bmatrix} 0{,}2092 & 1 \\ -0{,}6668 & 1 \\ 0{,}4574 & 1 \end{bmatrix} = \begin{bmatrix} 0{,}6976 & 0 \\ 0 & 3 \end{bmatrix}$$

$$(PP^T)^{-1} = \begin{bmatrix} 1{,}4335 & 0 \\ 0 & 0{,}3333 \end{bmatrix}$$

$$P^T(PP^T)^{-1}P = \begin{bmatrix} 0{,}2092 & 1 \\ -0{,}6668 & 1 \\ 0{,}4574 & 1 \end{bmatrix} \times \begin{bmatrix} 1{,}4335 & 0 \\ 0 & 0{,}3333 \end{bmatrix} \times \begin{bmatrix} 0{,}2092 & -0{,}6668 & 0{,}4574 \\ 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 0{,}3961 & 0{,}1333 & 0{,}4706 \\ 0{,}1333 & 0{,}9706 & -0{,}1039 \\ 0{,}4706 & -0{,}1039 & 0{,}6333 \end{bmatrix}$$

$$C_P = [I - P^T(PP^T)^{-1}P]\,\overline{C}^T = \left(\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} - \begin{bmatrix} 0{,}3961 & 0{,}1333 & 0{,}4706 \\ 0{,}1333 & 0{,}9706 & -0{,}1039 \\ 0{,}4706 & -0{,}1039 & 0{,}6333 \end{bmatrix} \right) \times \begin{bmatrix} 0 \\ 0{,}6668 \\ -0{,}4574 \end{bmatrix} = \begin{bmatrix} 0{,}1263 \\ -0{,}0279 \\ -0{,}0984 \end{bmatrix}$$

$$\|C_P\| = \sqrt{(0{,}1263)^2 + (-0{,}0279)^2 + (-0{,}0984)^2} = 0{,}1625$$

$$Y_{nov} = X_0 - \alpha r \frac{C_P}{\|C_P\|} = \begin{bmatrix} 1/3 \\ 1/3 \\ 1/3 \end{bmatrix} - \left(\frac{\frac{2}{9} \times \frac{1}{\sqrt{6}}}{0{,}1625} \right) \times \begin{bmatrix} 0{,}1263 \\ -0{,}0279 \\ -0{,}0984 \end{bmatrix} = \begin{bmatrix} 0{,}2628 \\ 0{,}3489 \\ 0{,}3883 \end{bmatrix}$$

$$D_2 Y_{nov} = \begin{bmatrix} 0{,}2092 & 0 & 0 \\ 0 & 0{,}3334 & 0 \\ 0 & 0 & 0{,}4574 \end{bmatrix} \times \begin{bmatrix} 0{,}2628 \\ 0{,}3489 \\ 0{,}3883 \end{bmatrix} = \begin{bmatrix} 0{,}0550 \\ 0{,}1163 \\ 0{,}1776 \end{bmatrix}$$

$$1 D_2 Y_{nov} = [1 \quad 1 \quad 1] \times \begin{bmatrix} 0{,}0550 \\ 0{,}1163 \\ 0{,}1776 \end{bmatrix} = 0{,}3489$$

$$X_3 = \frac{D_2 Y_{nov}}{1 D_2 Y_{nov}} = \frac{1}{0{,}3489} \times \begin{bmatrix} 0{,}0550 \\ 0{,}1163 \\ 0{,}1776 \end{bmatrix} = \begin{bmatrix} 0{,}1576 \\ 0{,}3334 \\ 0{,}5090 \end{bmatrix}$$

$$Z = C^T X_3 = [0 \quad 2 \quad -1] \times \begin{bmatrix} 0{,}1576 \\ 0{,}3334 \\ 0{,}5090 \end{bmatrix} = \mathbf{0{,}1578}$$

Ta postopek ponavljamo dokler ne pridemo do optimalne rešitve ali zadostnega približka.

Poglavje 5 Vhodni podatki

5.1 NETLIB – A collection of mathematical software, papers and databases

Netlib je repozitorij, ki vsebuje prosto dostopno programsko opremo, dokumente in podatkovne zbirke za interese numeričnega, znanstvenega računanja, ter podobne skupnosti. Repozitorij vzdržuje AT&T Bell Laboratories na univerzi v Tennessee-ju in Oak Ride National Laboratoty, ter posamezni sodelavci širom sveta. Zbirka je replicirana na večih spletnih straneh po svetu in se samodejno posodablja. Tako zagotavlja učinkovito storitev globalni skupnosti [27].

Zbirka testnih optimizacijskih problemov, ki izvira iz Netlib-a, je bila prenesena iz druge strani, kjer je podana v nekompresiranem MPS formatu [25]. Ta zbirka vsebuje probleme, kjer so rešitve spremenljivk realna števila.

5.2 MIPLIB – Mixed Integer Problem Library

MPLIB je knjižnica podaktov, ki je nastala zaradi potrebe raziskovalcev, ki so potrebovali testne podatke za testiranje programov, kateri rešujejo probleme z mešanimi števili in so najboljši možni približek problemov, ki jih srečujemo v realnem svetu [10].

Prva različica te knjižnice je nastala leta 1992 in je bila elektronsko dostopna. Vsebovala je probleme s celimi in realnimi števili kot tudi takšne s samo celimi števili. Od prve različice do danes je bila ta knjižnica posodobljena že petkrat. Zadnja različica je bila narejena leta 2010 in vsebuje 361 različnih problemov. Problemi so razdeljeni na tri skupine: lahki, težki in nerešljivi. Problem je opredeljen kot lahek, če je rešljiv v okviru ene ure. Težek, kadar je rešljiv, vendar ne v določenem časovnem okvirju. Z vsakim letom postane vedno več problemov rešljivih.

Odkar je MIPLIB bila predstavljena, je postala standard, za performančne primerjalne teste med programi, ki se osredotočajo na optimizacijo takšnih problemov.

5.3 Format vhodnih podatkov

MPS (Mathematical Programming System) je format datoteke, zapisan v ASCII standardu, za predstavitev problemov linearnega programiranja in programiranja z mešanimi števili.

Ta format je stolpično orientiran in vse komponente modela (spremenljivke, vrstice, itd.) imajo določeno svoje ime. Gre za starejši format, ki je nastavljen po sistemu luknjastih kartic: polja se pričnejo s stolpci 2, 5, 15, 25, 40 in 50. Odseki MPS datoteke so poimenovane s tako imenovanimi kazalniki (indicators), ki so zapisani v prvem stolpcu, imena komponent pa lahko izbiramo po lastni presoji, saj samemu programu za reševanje to ni pomembno. Iz zgodovine se je ohranilo, da v MPS datoteko zapisujemo z velikimi črkami, čeprav zdaj že veliko programov sprejema tudi male črke [11].

Vsebuje sedem odsekov oz. kazalnikov [12]:

- NAME — Poljubno ime problema, ki je zapisano v drugem imenskem polju in se začne v 15. stolpcu.

- ROWS — Tu so definirana imena vseh omejitev (vrstic), ki jih problem vsebuje. Ime omejitve je zapisano v prvem imenskem polju, ki se začne v 5. stolpcu. Poleg imena imajo določen še tip, ki je zapisan v 2. stolpcu. Vrednosti tipa:
  E – enakost (=),
  L – manjše kot (<=),
  G – večje kot (>=),
  N – ni omejitev (privzeto je to kriterijska funkcija).
  Zaporedje v katerem so določene omejitve navedene, ni pomembno.

- COLUMNS — V tem delu so podani neničelni koeficienti, vsebovani v matriki omejitev. V prvem imenskem polju je določen stolpec, drugo imensko polje pa mora ustrezati imenu vrstice. V prvem numeričnem polju (25. stolpec) je zapisana vrednost koeficienta, ki ustreza dani vrstici in stolpcu. V primeru, da je še kakšen koeficient v trenutnem stolpcu, se lahko zapis ponovi, tako da se poda v 40. stolpcu ime vrstice in v 50. stolpcu ustrezen koeficient.
  Tu se lahko nahaja še dodatna omejitev, ki označuje skupino spremenjivk, katerih rešitev mora biti celo število. To se uporablja pri problemih z mešanimi števili. Za začetek takšne skupine je v drugem stolpcu podano ime omejitve (npr. MARK000), nato v

tretjem stolpcu rezervirana beseda v narekovajih 'MARKER', ter v petem stolpcu 'INTORG'.
Določiti je treba tudi konec. Ta se določi enako kot začetek, le da v petem stolpcu 'INTORG' zamanjamo z 'INTEND'.

- RHS — Tukaj so podane vrednosti, ki jih vsebujejo omejitve na desni strani (RHS = right hand side oz. desna stran enačbe). V prvem imenskem polju je zapisan notranji identifikator, ki je največkrat kar RHS, v drugem imenskem polju pa je zapisano ime vrstice. V prvem numeričnem polju pa je zapisana vrednost desne strani omejitve v dani vrstici.

- RANGES — Opcijski del, kjer so lahko definirane omejitve tipa L in G hkrati. V prvem imenskem polju je zapisan notranji identifikator, ki je največkrat kar RANGE, v drugem imenskem polju pa je zapisana omejitev, ki je že poimenovana v odseku ROWS. Prva numerična vrednost vsebuje vrednost razpona. Če je na primer omejitev tipa G podana z vrednostjo desne strani 30 in vrednostjo razpona 10, to pomeni, da ima omejitev razpon [30, 40].

- BOUNDS — Opcijski del, kjer so podane meje posameznih spremenljivk. Meja je lahko tipa:
  UP – zgornja meja,
  LO – spodnja meja,
  FX – fiksne vrednosti (zgornja in spodnja meja je enaka),
  FR – proste spremenljivke (spodnja -∞, zgornja +∞),
  PL – nenegativne spremenljivke (zgornja meja +∞),
  MI – nepozitivne spremenljivke (spodnja meja -∞),
  BV – binarne spremenljivke.

  V prvem imenskem polju je podan tip meje, v drugem identifikator meje, v tretjem interni identifikator in vrednost v prvem numeričnem polju.

- ENDATA — Zadnji kazalnik, ki označuje konec datoteke.

5.3.1 Predloga MPS datoteke

```
Polja:      1          2          3          4          5          6
Stolpci:   2-3        5-12      15-22      25-36      40-47      50-61

           NAME                 problem name

           ROWS

            type      name

           COLUMNS
                     column      row       value      row       value
                     name        name                 name
           RHS
                     rhs         row       value      row       value
                     name        name                 name
           RANGES
                     range       row       value      row       value
                     name        name                 name
           BOUNDS

            type     bound       column     value
                     name        name
           ENDATA
```

5.3.2 Primer MPS datoteke [21]

Kriterijska funkcija: $Z = x_1 + 2*x_5 - x_8$

Omejitve:

$$2.5 <= 3*x_1 + x_2 - 2*x_4 - x_5 - x_8$$
$$2*x_2 + 1.1*x_3 <= 2.1$$
$$x_3 + x_6 = 4.0$$
$$1.8 <= 2.8*x_4 - 1.2*x_7 <= 5.0$$
$$3.0 <= 5.6*x_1 + x_5 + 1.9*x_8 <= 15.0$$

Kjer velja:

$$2.5 <= x_1$$
$$0 <= x_2 <= 4.1$$
$$0 <= x_3$$
$$0 <= x_4$$
$$0.5 <= x_5 <= 4$$
$$0 <= x_6$$
$$0 <= x_7$$
$$0 <= x_8 <= 4.3$$

Vsebina MPS datoteke

```
NAME          PRIMER1
ROWS
 N  OBJ
 G  ROW01
 L  ROW02
 E  ROW03
 G  ROW04
 L  ROW05
COLUMNS
    COL01     OBJ                    1.0
    COL01     ROW01             3.0  ROW05             5.6
    COL02     ROW01             1.0  ROW02             2.0
    COL03     ROW02             1.1  ROW03             1.0
    COL04     ROW01            -2.0  ROW04             2.8
    COL05     OBJ                    2.0
    COL05     ROW01            -1.0  ROW05             1.0
    COL06     ROW03             1.0
    COL07     ROW04            -1.2
    COL08     OBJ                   -1.0
    COL08     ROW01            -1.0  ROW05             1.9
RHS
    RHS1      ROW01             2.5
    RHS1      ROW02             2.1
    RHS1      ROW03             4.0
    RHS1      ROW04             1.8
    RHS1      ROW05            15.0
RANGES
    RNG1      ROW04             3.2
    RNG1      ROW05            12.0
BOUNDS
 LO BND1      COL01             2.5
 UP BND1      COL02             4.1
 LO BND1      COL05             0.5
 UP BND1      COL05             4.0
 UP BND1      COL08             4.3
ENDATA
```

Poglavje 6 Meritve

Meritve in integracija knjižnic za optimizacijo problemov linearnega programiranja so bile izvedene v okolju Net in programskem jeziku C# z uporabo programskega orodja Visual Studio 2010. Uporabljena sta bila tudi simpleksni in Karmarkar-jev algoritmem, ki sta bila sprogramirana in implementirana po postopkih, ki so opisani v prejšnjih poglavij. Vendar pa je pri večjih testnih množicah, ki so bile uporabljene za meritve, prišlo do numerične nestabilnosti. Zaradi tega dobljeni rezultati niso bili uporabljeni za analizo.

6.1 Orodja in vhodni podatki

Za pridobitev izmerjenih rezultat so bila uporabljena tri namenska programska orodja. Eno odprtokodno (lp_solve) ter dve komercialni (Gurobi in CPLEX).

6.1.1 lp_solve

Lp_solve je zastonjska odprtokodna programska oprema za reševanje problemov linearnega programiranja in je na voljo pod licenco LGPL. Rešuje tudi probleme z mešanimi števili. Podpira branje datotek v formatu LP in MPS. Klici se lahko izvajajo iz različnih programskih jezikov in okolij kot so: C, VB, .NET, Delphi, Excel, Java itd. Lp_solve nima omejitve glede velikosti vhodnega problema [28].

6.1.2 Gurobi optimizer

Gurobi optimizer je komercialno programsko orodje za linearno programiranje, kvadratno programiranje, kvadratno programiranje z omejitvijo, linearno programiranje z mešanimi števili in kvadratno programiranje z omejitvijo in mešanimi števili. Gurobi je dobil ime po svojih ustanoviteljih: Zonghao Gu, Edward Rothberg in Robert Bixby. Bixby je bil ustanovitelj CPLEX-a, medtem ko sta Rothberg in Gu vodila njegovo razvojno ekipo skoraj desetletje.
Nudi podporo za različne programske in modelirne jezike:

- objektni vmesnik: C++, Java, .NET in Python;
- matrični vmesnik: C, MATLAB in R;
- povezavo s standardnimi modelirnimi jeziki: AIMMS, AMPL, GAMS in MPL.

Zastonj je na voljo poskusna različica, ki obsega polno funkcionalnost, vendar pa ima omejitev pri velikosti problemov, ki jih lahko rešujemo [23, 29].

6.1.3 IBM ILOG CPLEX optimization studio

CPLEX je paket programov za optimizacijo. Leta 2004 si je prislužil INFORMS [31] Impact Prize [32]. Ime je dobil po simpleksni metodi, ki je bila implementirana v programskem jeziku C. Danes podpira več optimizacijskih metod in različnih vmesnikov za programske jezike. CPLEX je bil prvič ponujen v komercialne namene leta 1988, takrat v lasti podjetja CPLEX Optimization Inc. Leta 1997 ga je prevzelo podjetje ILOG, januarja 2009 pa IBM. Do danes se še vedno aktivno razvija pod okriljem podjetja IBM.
Uporablja se za reševanje zelo velikih problemov linearnega programiranja. Za doseganje optimalnih rešitev uporablja različice primarne in dualne simpleksne metode, metodo notranjih točk s pregradami (ang. barrier interior point method). Rešuje tudi probleme konveksnega in nekonveksnega kvadratnega programiranja, kot tudi probleme konveksnega kvadratnega programiranja z omejitvami.
Prav tako ponuja tudi vmesnike za različna programska okolja in jezike: C++, C#, Java, Python in modelirne jezike: AIMMS, AMPL, GAMS, MPL, OpenOpt, OptimJ in TOMLAB [33].
Enako kot Gurobi je tudi za CPLEX zastonj dosegljiva poskusna različica, ki obsega polno funkcionalnost in ima velikostno omejitev problemov [22].

6.1.4 Vhodni podatki

ki so podani v tabeli, so ključnega pomena. Glede na te podatke bo tudi izvedena analiza rezultatov. Podatki v tabeli [30]:

- NAME – ime problema;
- ROWS – število vrstic oz. število podanih omejitev (neenakosti);
- COLS – število stolpcev oz. število podanih spremenljivk;
- NONZEROS – število neničelnih koeficientov;
- BYTES – velikost celotnega problema (velikost datoteke v bajtih);
- OPTIMAL VALUE – optimalna rešitev problema.

NAME	ROWS	COLS	NONZEROS	BYTES	OPTIMAL VALUE
25FV47	822	1571	11127	70477	5.5018458883E+03
80BAU3B	2263	9799	29063	298952	9.8723216072E+05
ADLITTLE	57	97	465	3690	2.2549496316E+05
AFIRO	28	32	88	794	-4.6475314286E+02
AGG2	517	302	4515	32552	-2.0239252356E+07
AGG3	517	302	4531	32570	1.0312115935E+07

BANDM	306	472	2659	19460	-1.5862801845E+02
BEACONFD	174	262	3476	17475	3.3592485807E+04
BLEND	75	83	521	3227	-3.0812149846E+01
BNL1	644	1175	6129	42473	1.9776292856E+03
BNL2	2325	3489	16124	127145	1.8112365404E+03
BOEING1	351	384	3865	25315	-3.3521356751E+02
BOEING2	167	143	1339	8761	-3.1501872802E+02
BORE3D	234	315	1525	13160	1.3730803942E+03
BRANDY	221	249	2150	14028	1.5185098965E+03
CAPRI	272	353	1786	15267	2.6900129138E+03
CYCLE	1904	2857	21322	166648	-5.2263930249E+00
CZPROB	930	3523	14173	92202	2.1851966989E+06
D2Q06C	2172	5167	35674	258038	1.2278423615E+05
D6CUBE	416	6184	43888	167633	3.1549166667E+02
DEGEN2	445	534	4449	24657	-1.4351780000E+03
DFL001	6072	12230	41873	353192	1.12664E+07
E226	224	282	2767	17749	-1.8751929066E+01
ETAMACRO	401	688	2489	21915	-7.5571521774E+02
FFFFF800	525	854	6235	39637	5.5567961165E+05
FINNIS	498	614	2714	23847	1.7279096547E+05
FIT1D	25	1026	14430	51734	-9.1463780924E+03
FIT1P	628	1677	10894	65116	9.1463780924E+03
FIT2D	26	10500	138018	482330	-6.8464293294E+04
FIT2P	3001	13525	60784	439794	6.8464293232E+04
FORPLAN	162	421	4916	25100	-6.6421873953E+02
GANGES	1310	1681	7021	60191	-1.0958636356E+05
GFRD-PNC	617	1092	3467	24476	6.9022359995E+06
GREENBEA	2393	5405	31499	235711	-7.2462405908E+07
GREENBEB	2393	5405	31499	235739	-4.3021476065E+06
GROW15	301	645	5665	35041	-1.0687094129E+08
GROW22	441	946	8318	50789	-1.6083433648E+08
GROW7	141	301	2633	17043	-4.7787811815E+07
ISRAEL	175	142	2358	12109	-8.9664482186E+05
KB2	44	41	291	2526	-1.7499001299E+03
LOTFI	154	308	1086	6718	-2.5264706062E+01
MAROS	847	1443	10006	65906	-5.8063743701E+04
MAROS-R7	3137	9408	151120	4812587	1.4971851665E+06
MODSZK1	688	1620	4158	40908	3.2061972906E+02
NESM	663	2923	13988	117828	1.4076073035E+07
PEROLD	626	1376	6026	47486	-9.3807580773E+03
PILOT	1442	3652	43220	278593	-5.5740430007E+02
PILOT.JA	941	1988	14706	97258	-6.1131344111E+03
PILOT.WE	723	2789	9218	79972	-2.7201027439E+06
PILOT4	411	1000	5145	40936	-2.5811392641E+03
PILOTNOV	976	2172	13129	89779	-4.4972761882E+03
QAP8	913	1632	8304	(see NOTES)	2.0350000000E+02
QAP12	3193	8856	44244	(see NOTES)	5.2289435056E+02
RECIPE	92	180	752	6210	-2.6661600000E+02
SC105	106	103	281	3307	-5.2202061212E+01

SC205	206	203	552	6380	-5.2202061212E+01
SC50A	51	48	131	1615	-6.4575077059E+01
SC50B	51	48	119	1567	-7.0000000000E+01
SCAGR25	472	500	2029	17406	-1.4753433061E+07
SCAGR7	130	140	553	4953	-2.3313892548E+06
SCFXM1	331	457	2612	19078	1.8416759028E+04
SCFXM2	661	914	5229	37079	3.6660261565E+04
SCFXM3	991	1371	7846	53828	5.4901254550E+04
SCORPION	389	358	1708	12186	1.8781248227E+03
SCRS8	491	1169	4029	36760	9.0429998619E+02
SCSD1	78	760	3148	17852	8.6666666743E+00
SCSD6	148	1350	5666	32161	5.0500000078E+01
SCSD8	398	2750	11334	65888	9.0499999993E+02
SCTAP1	301	480	2052	14970	1.4122500000E+03
SCTAP2	1091	1880	8124	57479	1.7248071429E+03
SCTAP3	1481	2480	10734	78688	1.4240000000E+03
SEBA	516	1028	4874	38627	1.5711600000E+04
SHARE1B	118	225	1182	8380	-7.6589318579E+04
SHARE2B	97	79	730	4795	-4.1573224074E+02
SHELL	537	1775	4900	38049	1.2088253460E+09
SHIP04L	403	2118	8450	57203	1.7933245380E+06
SHIP04S	403	1458	5810	41257	1.7987147004E+06
SHIP08L	779	4283	17085	117083	1.9090552114E+06
SHIP08S	779	2387	9501	70093	1.9200982105E+06
SHIP12L	1152	5427	21597	146753	1.4701879193E+06
SHIP12S	1152	2763	10941	82527	1.4892361344E+06
SIERRA	1228	2036	9252	76627	1.5394362184E+07
STAIR	357	467	3857	27405	-2.5126695119E+02
STANDATA	360	1075	3038	26135	1.2576995000E+03
STANDMPS	468	1075	3686	29839	1.4060175000E+03
STOCFOR1	118	111	474	4247	-4.1131976219E+04
STOCFOR2	2158	2031	9492	79845	-3.9024408538E+04
STOCFOR3	16676	15695	74004	(see NOTES)	-3.9976661576E+04
TRUSS	1001	8806	36642	(see NOTES)	4.5881584719E+05
TUFF	334	587	4523	29439	2.9214776509E-01
VTP.BASE	199	203	914	8175	1.2983146246E+05
WOOD1P	245	2594	70216	328905	1.4429024116E+00
WOODW	1099	8405	37478	240063	1.3044763331E+00

Tabela 4 – Seznam optimizacijskih problemov iz podatkovne zbirke Netlib.

6.2 Rezultati meritev

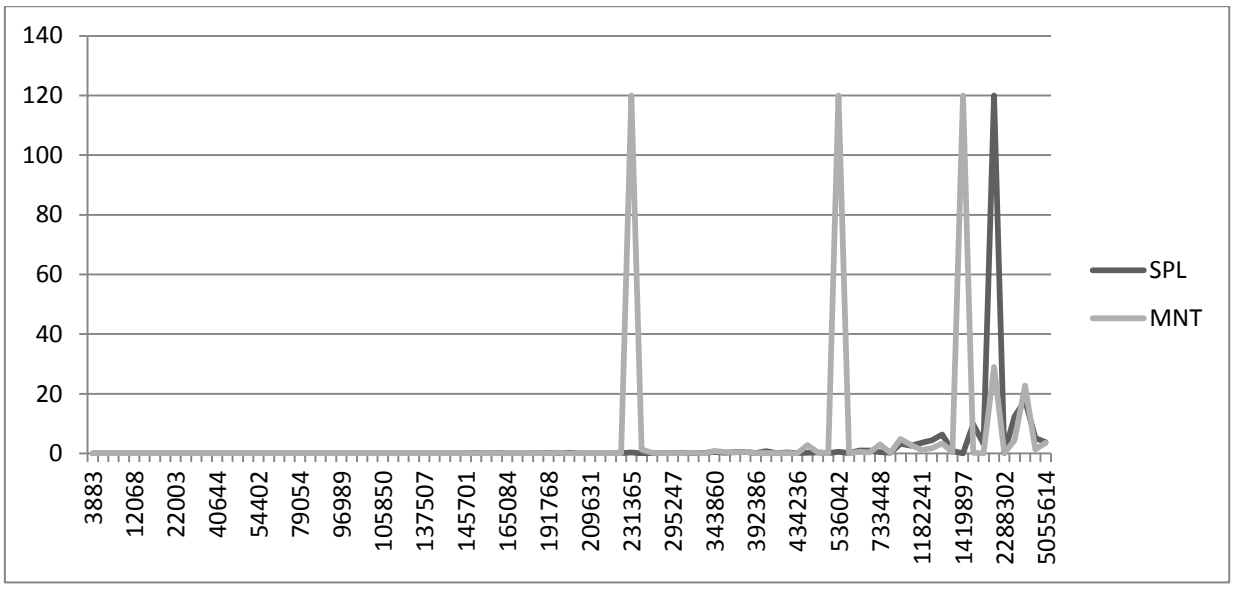
Rezultati meritev so prikazani v grafih, kjer primerjamo dve različni metodi za optimizacijo z uporabo orodij lp_solve, Gurobi in CPLEX. Uporabljeni sta tudi zbirki problemov Netlib in Miplib. Časovna enota na grafih je vedno podana v sekundah.

Oznake na grafih:

- SPL – simpleksna metoda;
- MNT – metoda notranjih točk.

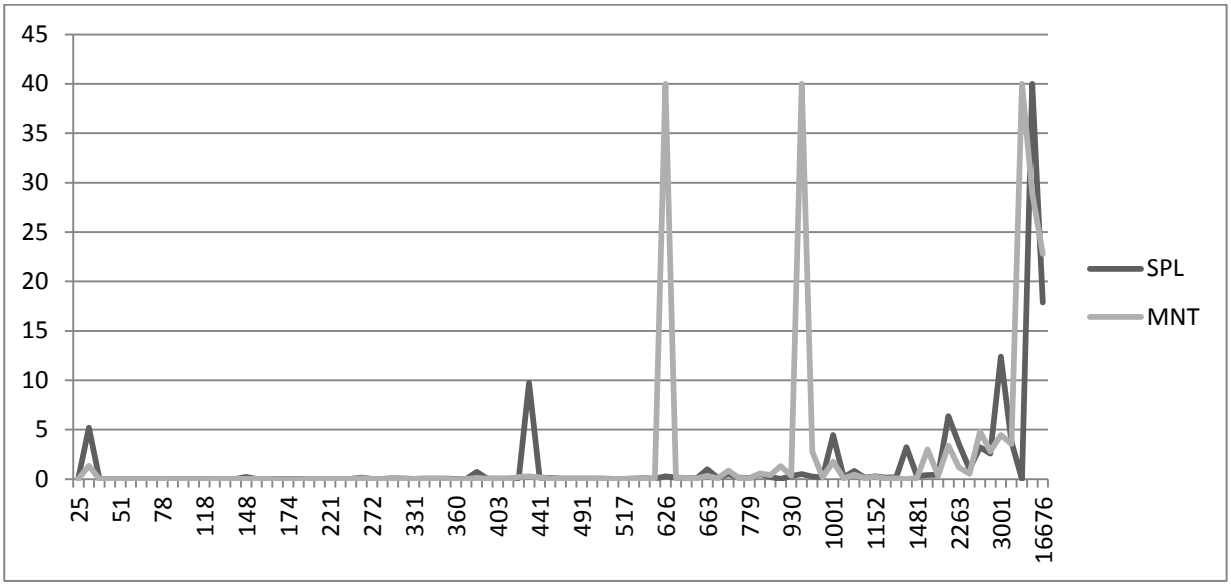
NETLIB in LP_SOLVE

Čas izvajanja algoritma v odvisnosti od velikosti problema.



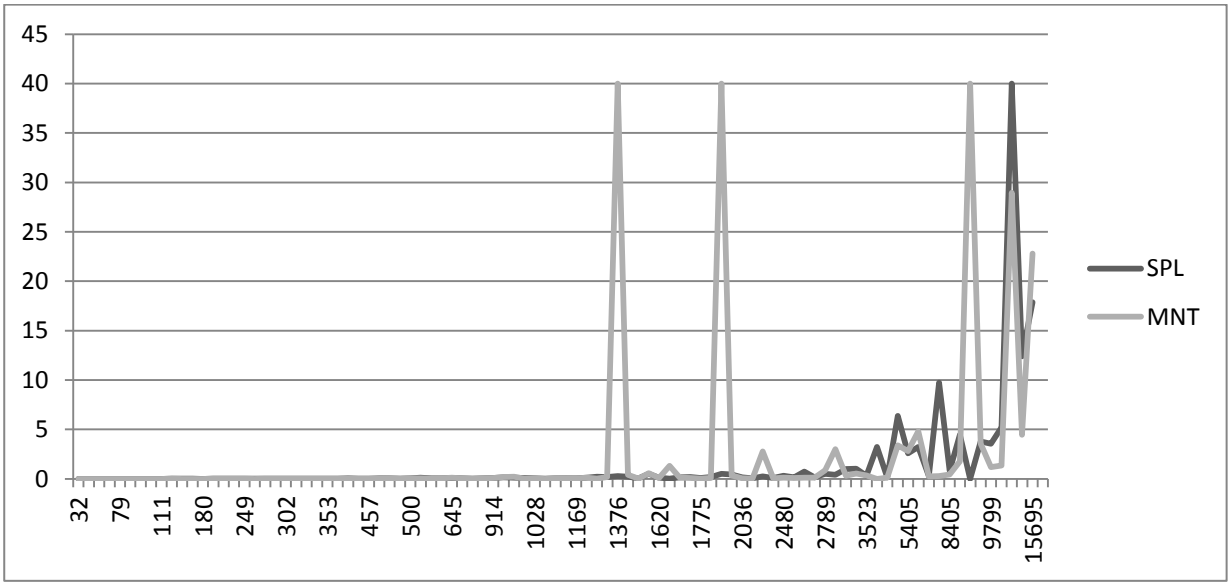
Graf 1

Čas izvajanja algoritma v odvisnosti od števila vrstic.



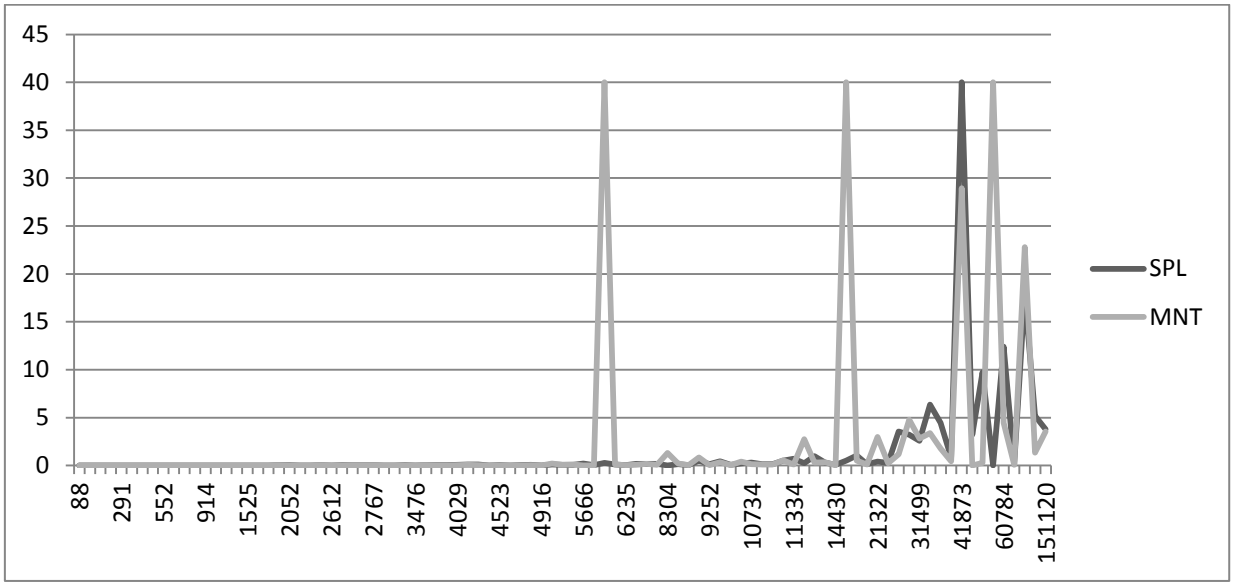
Graf 2

Čas izvajanja algoritma v odvisnosti od števila stolpcev.



Graf 3

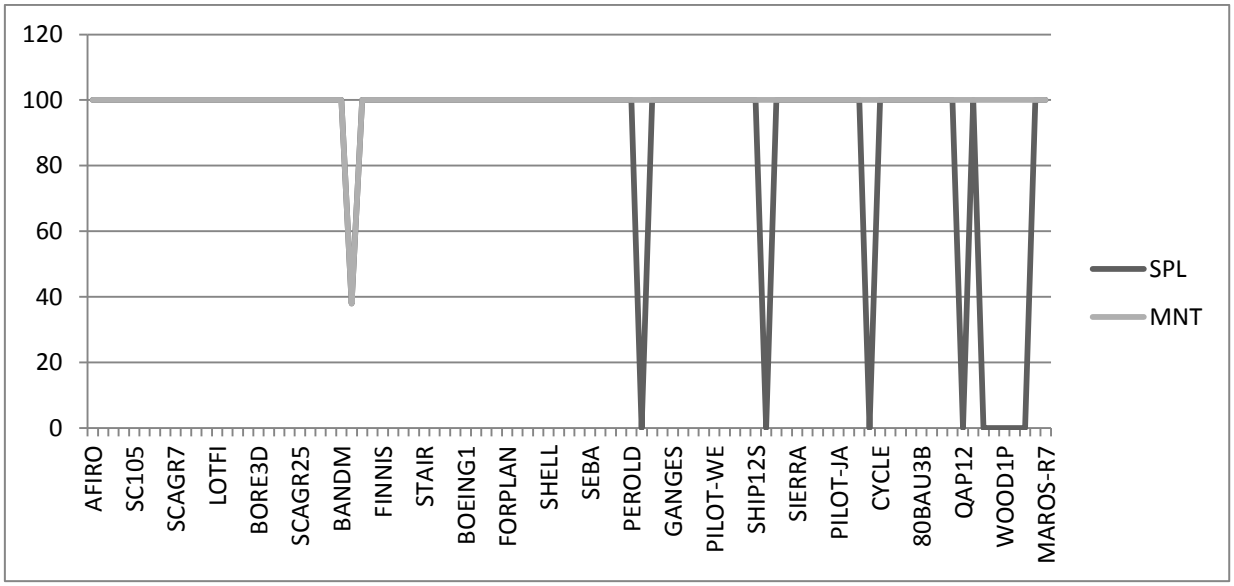
Čas izvajanja algoritma v odvisnosti od števila neničelnih koeficientov.



Graf 4

Pravilen izračun problema v procentih. Problemi, ki so ostali nerešeni ali so odstopali za več kot 100 %, so označeni z 0.

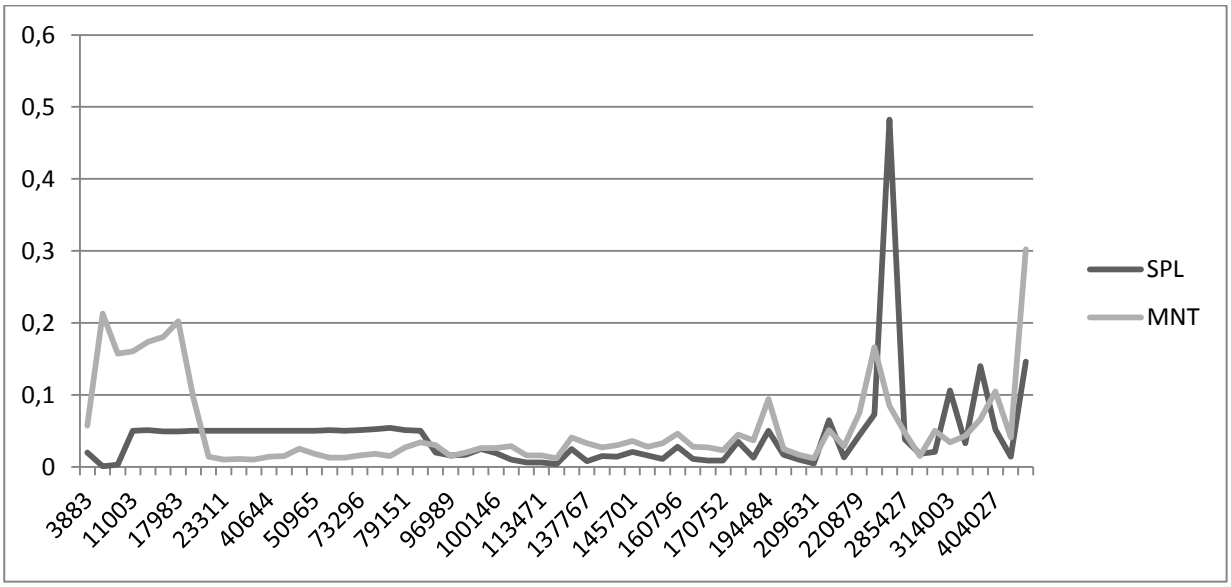
- Primer 1: Pravilen rezultat problema je 42, dobili pa smo končen rezultat 90. Absolutna vrednost razlike med optimalnim in dobljenim rezultatom je 48, kar znaša približno 114 % optimalne vrednosti. Na grafu ima vrednost 0.
- Primer 2: Dobimo končni rezultat 16, kar je 38% od optimalne vrednosti. Na grafu ima vrednost 38.
- Primer 3: Rešitev je enaka optimalni vrednosti. Vrednost na grafu je 100.



Graf 5

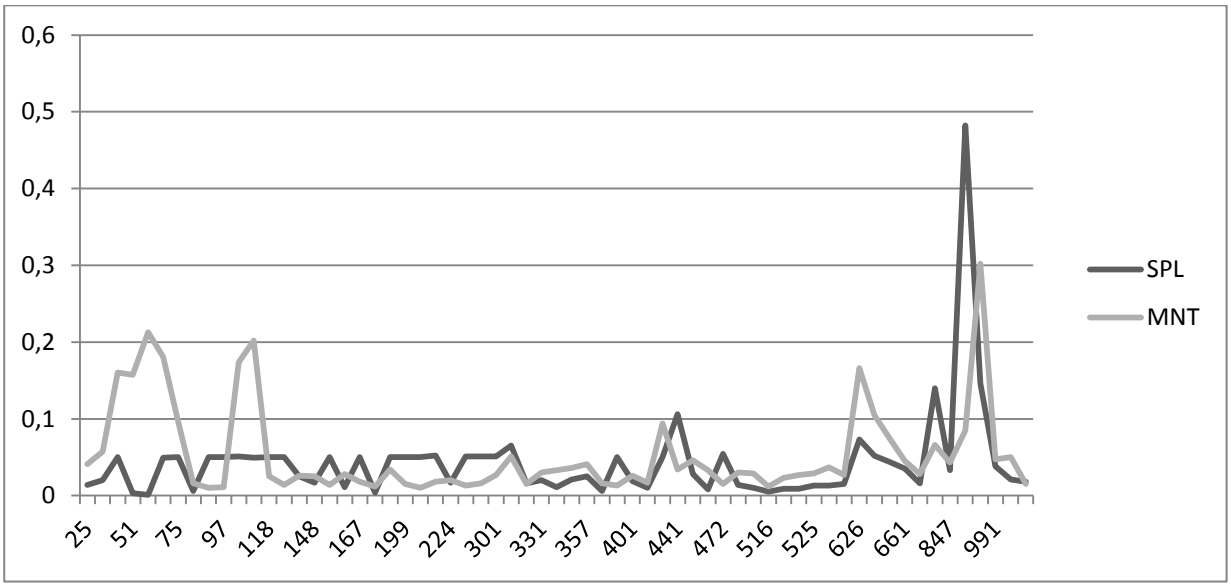
NETLIB in GUROBI

Čas izvajanja algoritma v odvisnosti od velikosti problema.



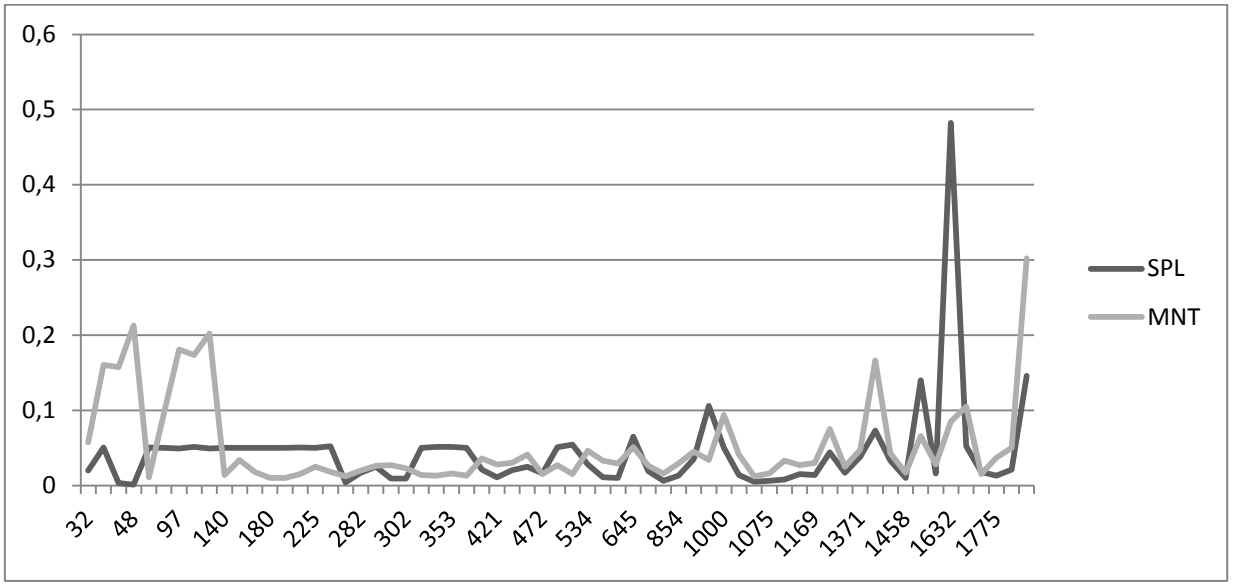
Graf 6

Čas izvajanja algoritma v odvisnosti od števila vrstic.



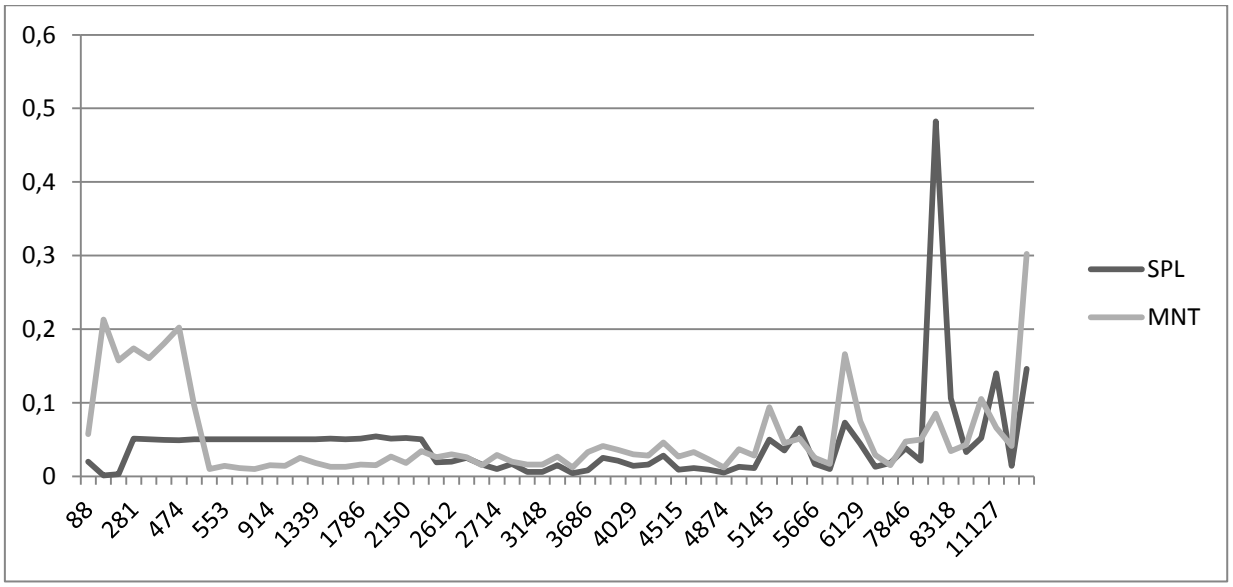
Graf 7

Čas izvajanja algoritma v odvisnosti od števila stolpcev.



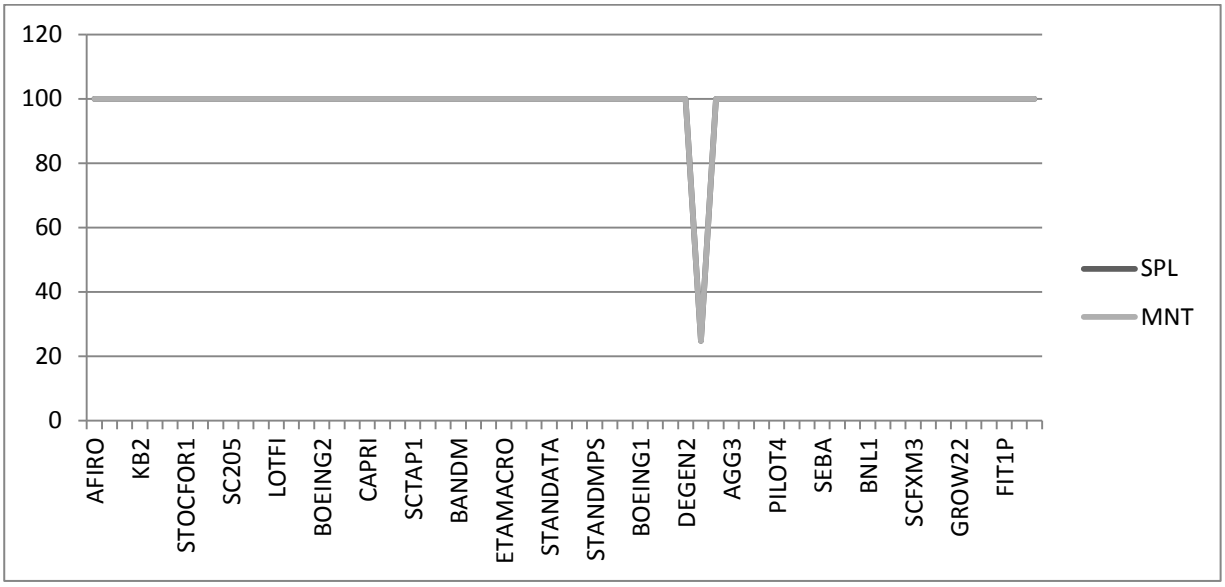
Graf 8

Čas izvajanja algoritma v odvisnosti od števila neničelnih koeficientov.



Graf 9

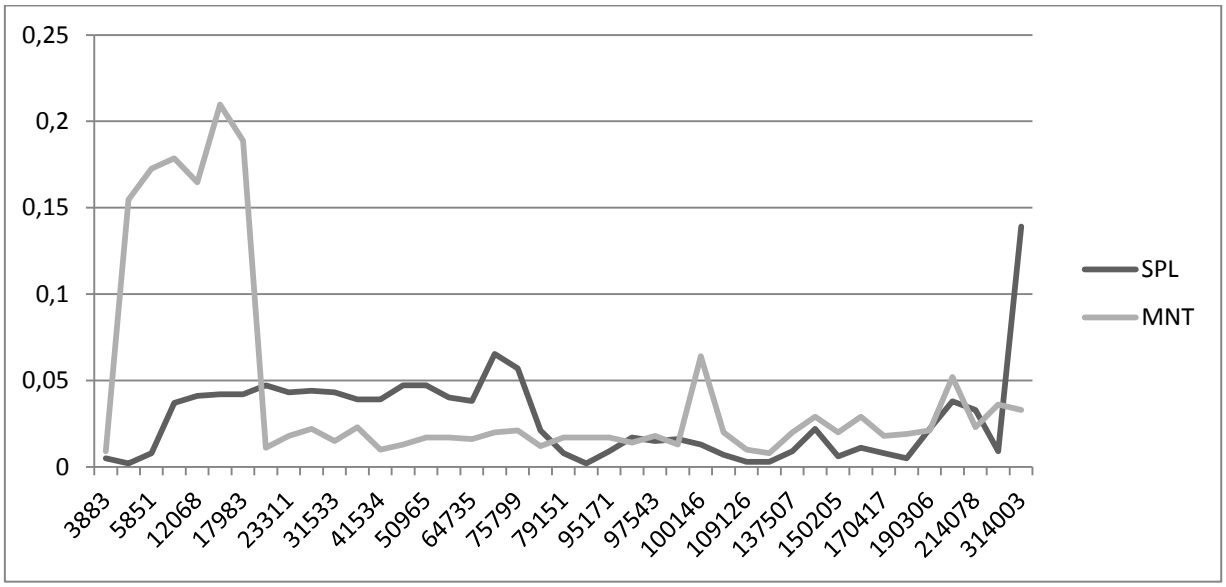
Dosežen pravilen izračun problemov v procentih.



Graf 10

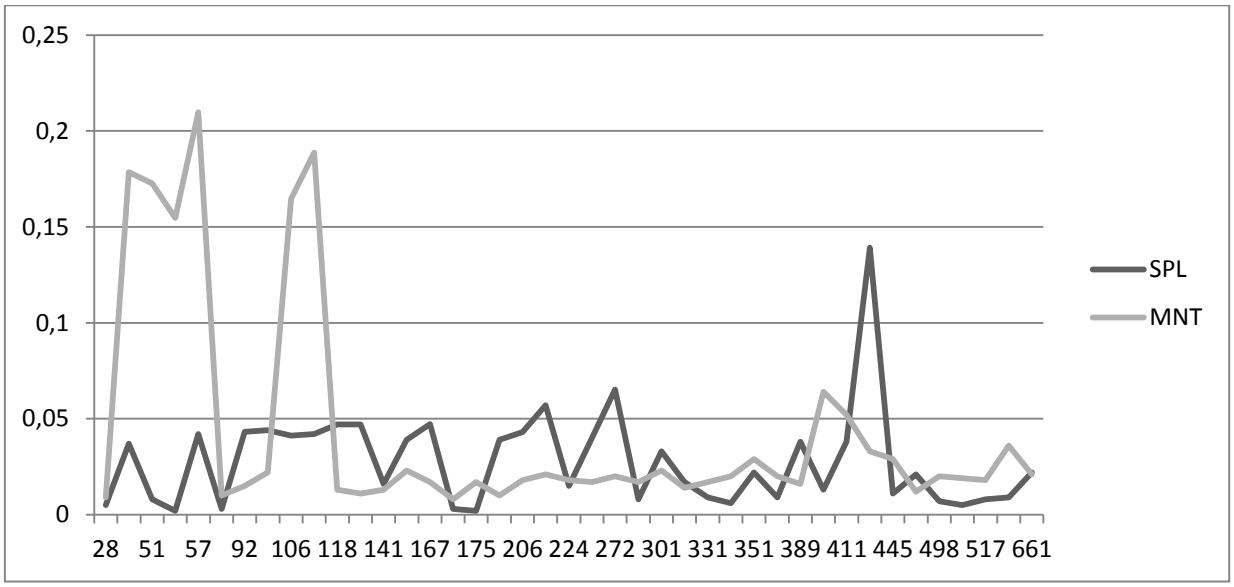
NETLIB in CPLEX

Čas izvajanja algoritma v odvisnosti od velikosti problema.



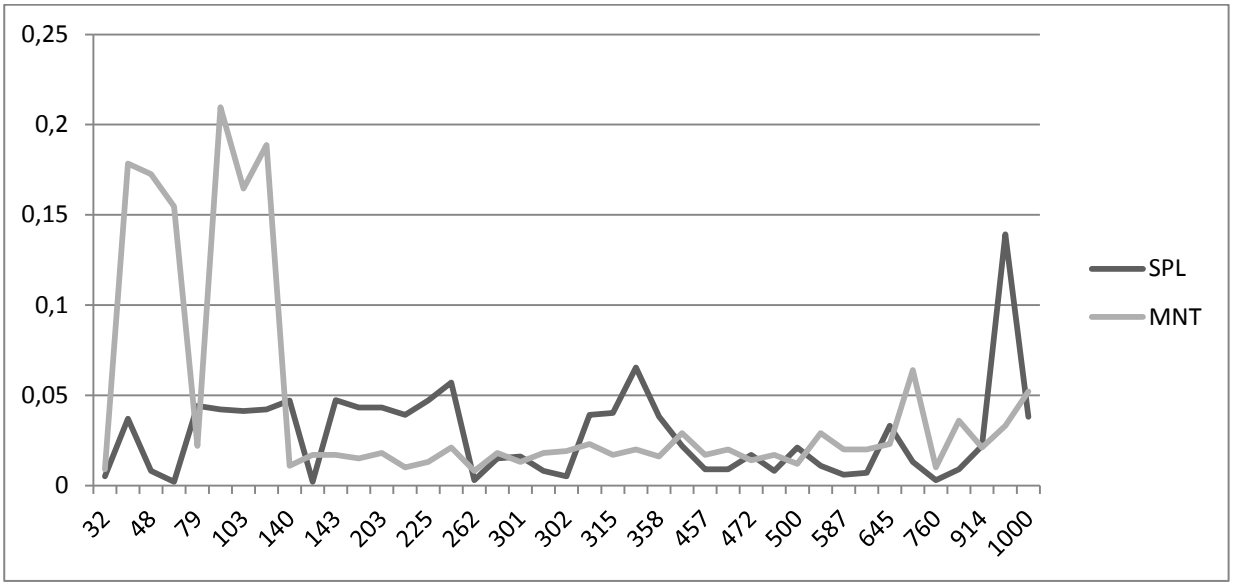
Graf 11

Čas izvajanja algoritma v odvisnosti od števila vrstic.



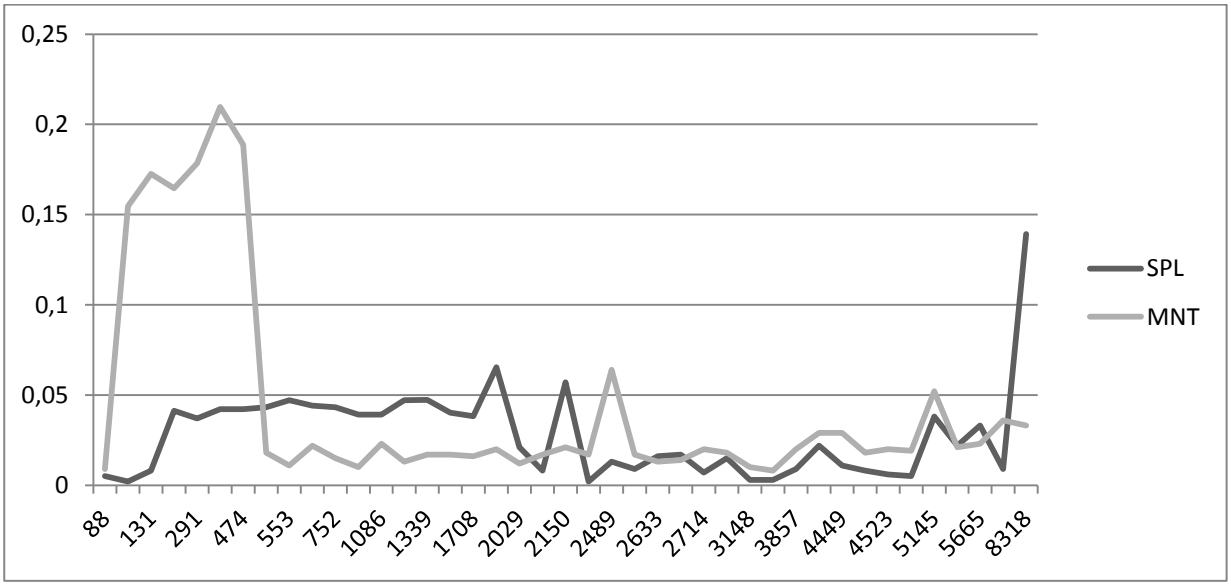
Graf 12

Čas izvajanja algoritma v odvisnosti od števila stolpcev.



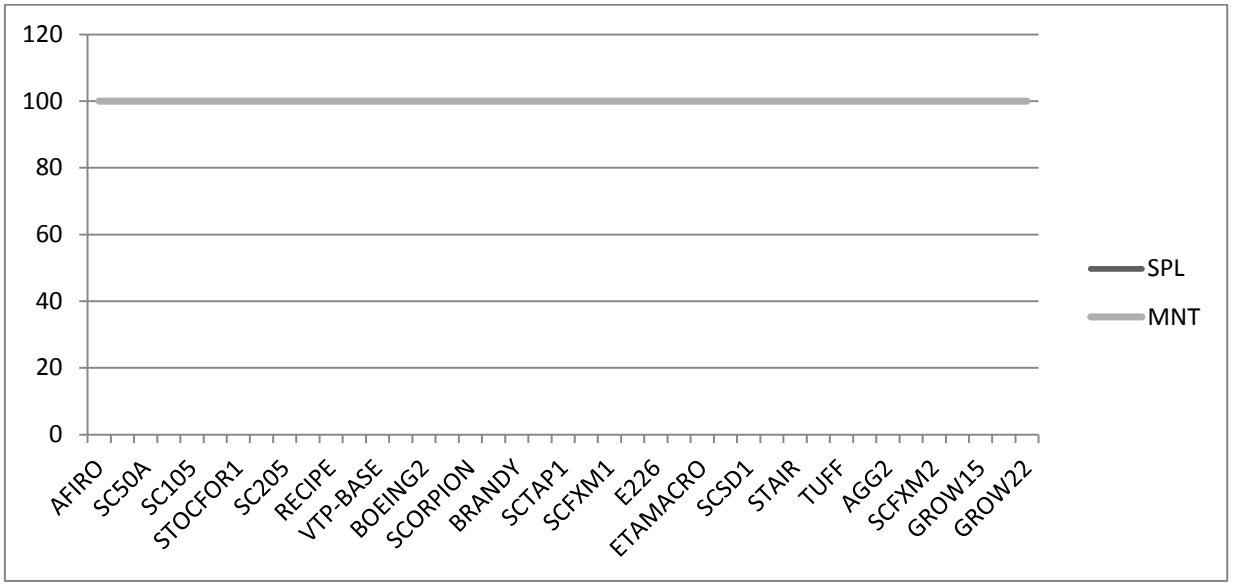
Graf 13

Čas izvajanja algoritma v odvisnosti od števila neničelnih koeficientov.



Graf 14

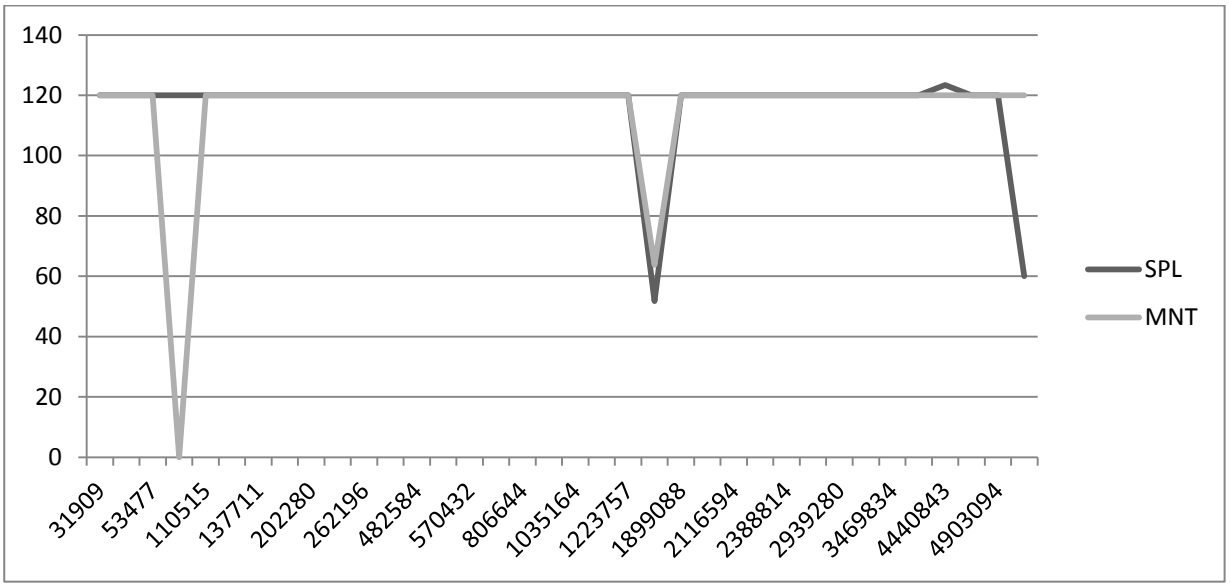
Dosežen pravilen izračun problemov v procentih.



Graf 15

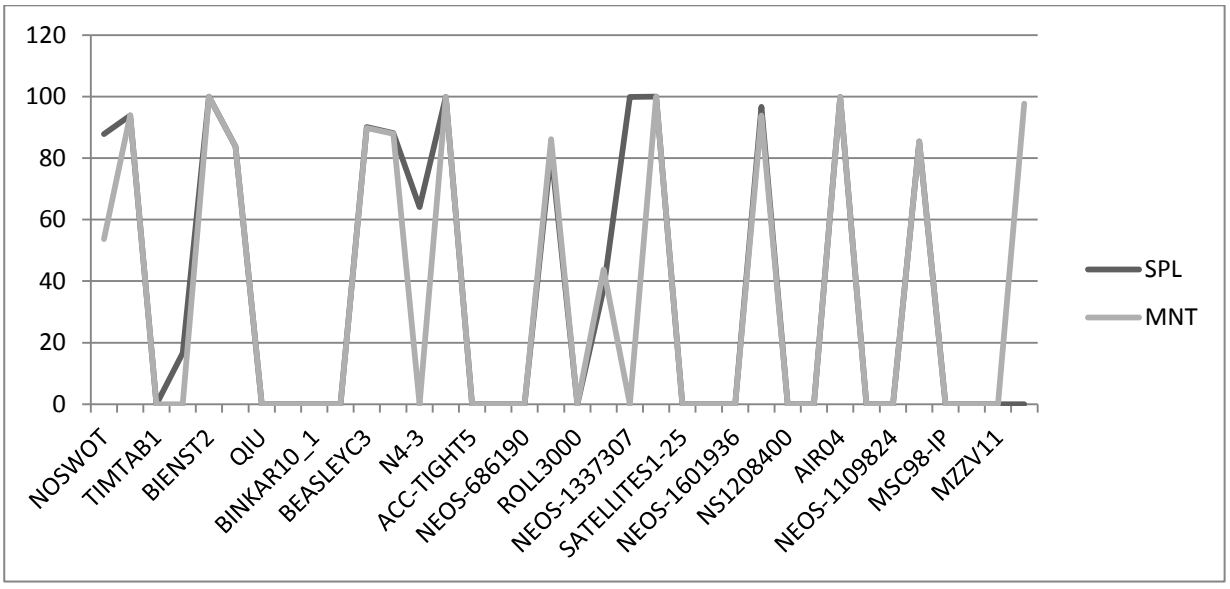
MIPLIB in LP_SOLVE

Čas izvajanja algoritma v odvisnosti od velikosti problema.



Graf 16

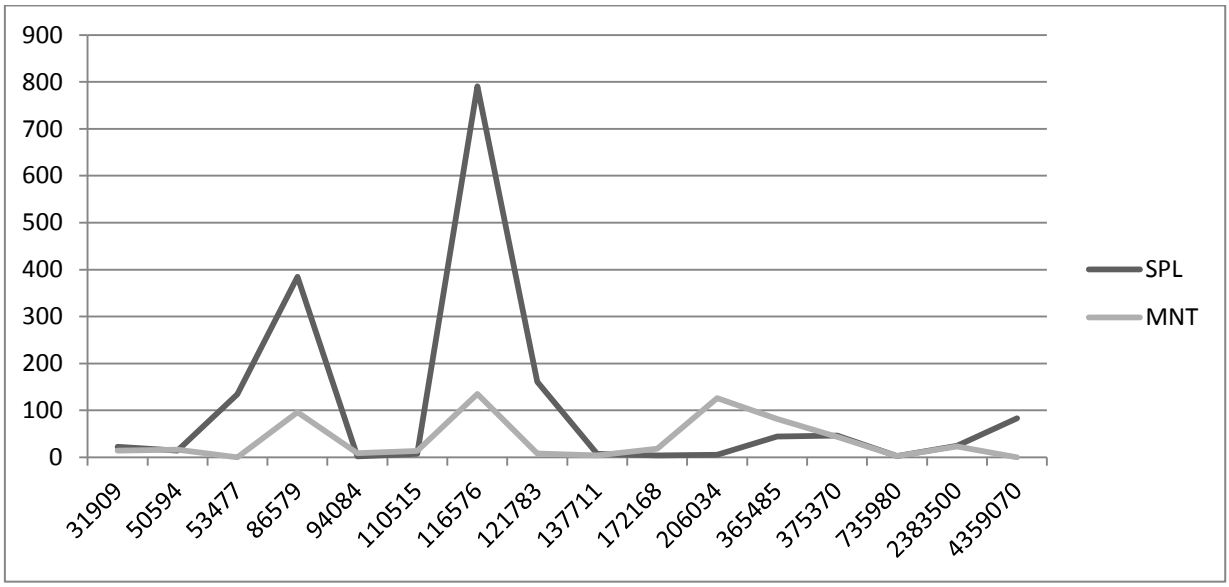
Dosežen pravilen izračun problemov v procentih.



Graf 17

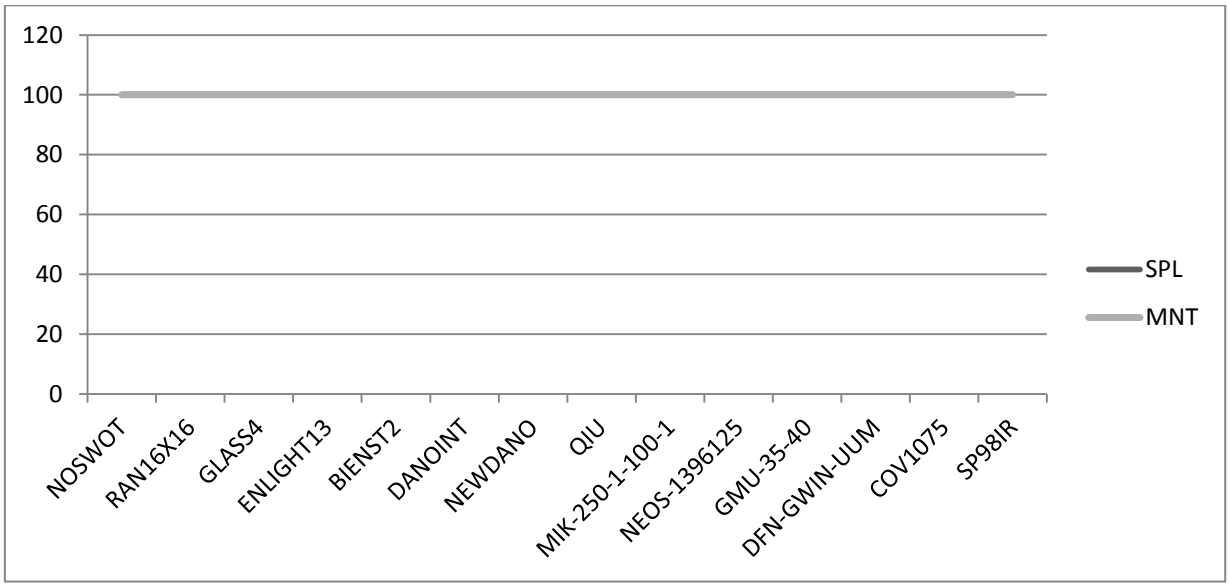
MIPLIB in GUROBI

Čas izvajanja algoritma v odvisnosti od velikosti problema.



Graf 18

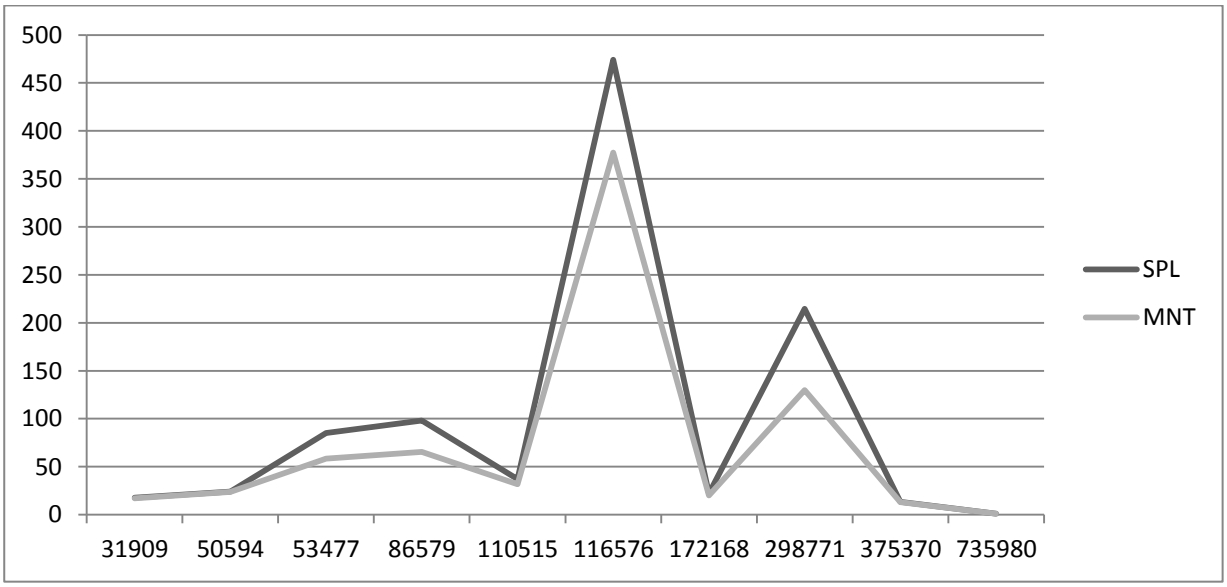
Dosežen pravilen izračun problemov v procentih.



Graf 19

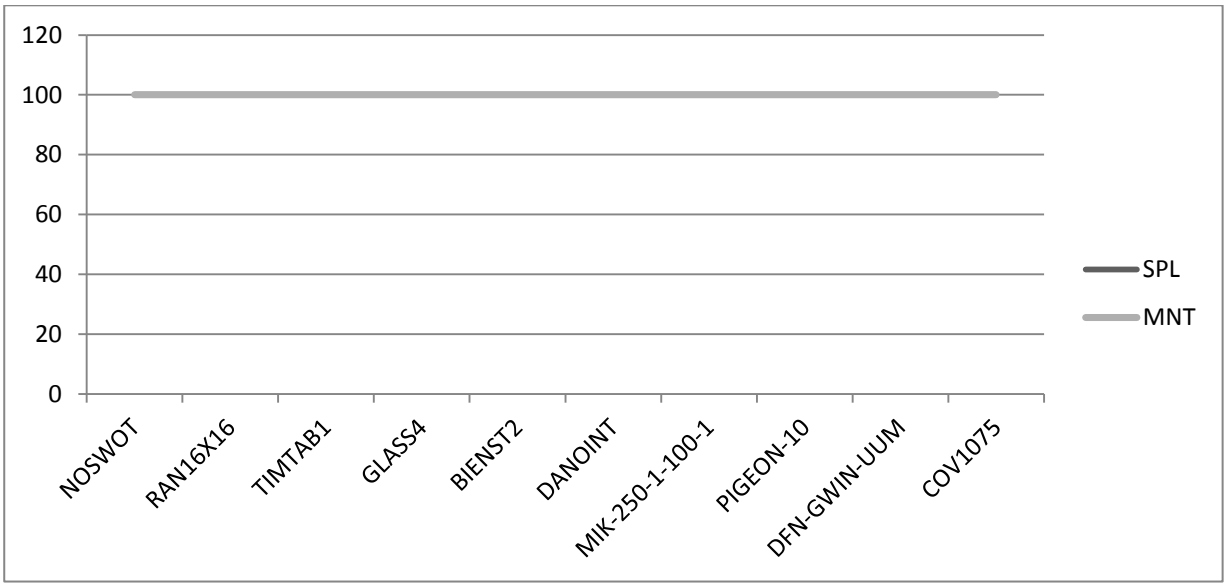
MIPLIB in CPLEX

Čas izvajanja algoritma v odvisnosti od velikosti problema.



Graf 20

Dosežen pravilen izračun problemov v procentih.



Graf 21

Poglavje 7 Zaključek

Spoznali smo nekatere načine reševanja problemov linearnega programiranja. Kako so nastali, se razvijali ter prilagajali potrebam. Pogledali smo si tudi primere, kjer smo uporabili predstavljene algoritme za reševanje optimizacijskih problemov. Največji poudarek je bil na dveh metodah, ki nas najbolj zanimata in sta danes zelo pogosto v uporabi: simpleksna metoda in metoda notranjih točk. Predvsem nas zanima, katera je hitrejša in učinkovitejša.

Če pogledamo rezultate meritev, opazimo, da se natančnost dobljenih rezultatov med zastonjskim odprtokodnim in plačljivima orodjema bistveno razlikuje. Predvsem pri zbirki problemov Miplib, kjer so optimizacijski problemi z mešanimi števili, ki zahtevajo, da so rešitve določenih spremenljivk cela števila. Problemi z mešanimi števili so težje rešljivi in potrebujejo implementacijo dodatnih algoritmov. Eden od takšnih algoritmov je predstavljen v poglavju 4: Gomory-jev algoritem rezanja ravnine. Kot je bilo omenjeno, ta algoritem zagotovi, da je rešitev celoštevilska. Obstajajo izboljšane različice tega algoritma, ki zagotovijo učinkovitost takšnih rešitev. To pa v tem primeru ni toliko pomembno, saj nas predvsem zanima primerjava simpleksne metode in metode notranjih točk.

V primeru, da zanemarimo razlike med posameznimi orodji in se osredotočimo samo na delovanje obeh metod, opazimo, da se v okviru posameznega orodja obnašata zelo podobno. Težko bi določili, katera metoda se je bolje obnesla. V nekaterih primerih se je metoda notranjih točk izkazala za nekoliko boljšo kakor simpleksna metoda. Eden takšnih primerov je viden na grafu Graf 18. Vendar, če pogledamo celotne rezultate, bi težko rekli, da je suverena zmagovalka. Obe metodi se obnašata podobno pri vseh postavljenih pogojih: velikost problema, število omejitev, število spremenljivk in število neničelnih koeficientov. Vsekakor pa obstajajo problemi, ki so precej bolje rešljivi z eno kakor z drugo metodo, vendar je to odvisno od samega tipa problema in ne njegove zunanje lastnosti, kot je velikost.

Sklepamo lahko, da je v primeru teh dveh algoritmov pomembnejša predvsem učinkovita implementacija posameznega algoritma, kakor pa izbira le-tega.

Literatura

[1] Jereb B., Skok D., Šafran M., Škornik M., Programi za logistike, LINDO – Optimizacija stroškov, 2014. Dosegljivo: http://bit.ly/1BvjmyP.

[2] Bergant B., Kratka zgodovina linearnega programiranja, 2007. Dosegljivo: http://bit.ly/1rLJiz5.

[3] Usenik J., Optimizacija logističnih procesov, študijsko gradivo. Fakulteta za logistiko, Celje, 2009.

[4] Mojca Berden, Zgodovina linearnega programiranja, 2009. Dosegljivo: http://bit.ly/1CUWkTN.

[5] Marijan Žura, Matematično programiranje, skripta, Fakulteta za gradbeništvo in geodezijo, Ljubljana, 2003. Dosegljivo: http://bit.ly/1xVM9OG.

[6] Elipsoidna metoda. *Wikipedia*, 2014. Dosegljivo: http://bit.ly/1xMS4FH.

[7] LPs in Polynomial Time, Intro to Combinatorial Optimizations. Brown Computer Science, Brown University, 2014. Dosegljivo: http://bit.ly/YzjDTv.

[8] Robert Sedgewick and Kevin Wayne, Algorithms and Data Structures. Department of Computer Science, Princeton University, 2007. Dosegljivo: http://bit.ly/1s0OXrh.

[9] Daniela Petkoviček, Matematičke metode u kemijskom inženjerstvu. Fakulteta za kemijsko inženerstvo in tehnologijo, 2007. Dosegljivo: http://bit.ly/ZhPxnz.

[10] MIPLIB, 2014. Dosegljivo: http://miplib.zib.de.

[11] MPS (format). *Wikipedia*, 2014. Dosegljivo: http://bit.ly/WMxjst.

[12] IBM, Records in MPS format, 2014. Dosegljivo: http://ibm.co/YziYBq.

[13] John von Neumann. *Wikipedia*, 2014. Dosegljivo: http://bit.ly/1eGaVCF.

[14] Interior point method. *Wikipedia*, 2014. Dosegljivo: http://bit.ly/1tMx6nT.

[15] Cutting–plane method. *Wikipedia*, 2014. Dosegljivo: http://bit.ly/ZhJsYj.

[16] F. D. Lewis, Cutting–plane Techniques. Department of Computer Science, University of Kentucky, 1999. Dosegljivo: http://bit.ly/1qGBnqq.

[17] Daniel Guetta, Gomory's Cutting-Plane method example. Columbia Business School, 2010. Dosegljivo: http://bit.ly/1qRR3G0.

[18] Nicole Lemire, An introduction to Karmarkar's method. University of Windsor, 1989.

[19] D. Nagesh Kumar, Other algorithms. Optimization methods, NPTEL, 2014. Dosegljivo: http://bit.ly/1usSBZq.

[20] D. Bertsimas, Optimization methods. MIT Sloan School of Management, 2009. Dosegljivo: http://bit.ly/1BupG9T

[21] Sample MPS Input Files, 2014. Dosegljivo: http://bit.ly/1qOcVoe.

[22] CPLEX Optimizer, 2014. Dosegljivo: http://ibm.co/1toVJFC

[23] Gurobi Optimizer, 2014. Dosegljivo: http://bit.ly/1miKY6P.

[24] Karmarkar's algorithm, 2014. Wikipedia. Dosegljivo: http://bit.ly/WQ9nVc.

[25] Computational optimization and applications software forum, A collection od optimization problems, 2014. Dosegljivo: http://bit.ly/1r2Pera.

[26] GeoGebra, 2014. Dosegljivo: http://www.geogebra.org/cms/en/.

[27] Netlib Repository, a collection of mathematical software, papers, and databases, 2014. Dosegljivo: http://www.netlib.org.

[28] Lp_solve, 2014. Dosegljivo: http://bit.ly/1mQOmWZ.

[29] Gurobi. *Wikipedia*, 2014. Dosegljivo: http://en.wikipedia.org/wiki/Gurobi.

[30] Problem summary table, 2004. Dosegljivo: http://bit.ly/1mR6FeM.

[31] INFORMS, 2014. Dosegljivo: http://bit.ly/1pcLXkC.

[32] Impact Prize, 2014. Dosegljivo: http://bit.ly/1spej1Z.

[33] CPLEX. *Wikipedia*, 2014. Dosegljivo: http://en.wikipedia.org/wiki/CPLEX.

Slike